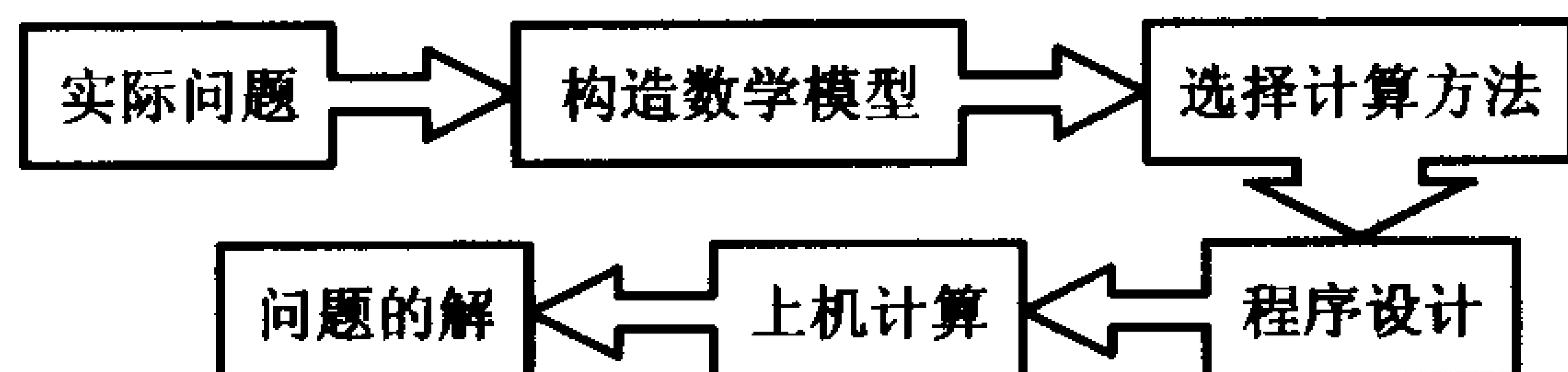
目录

第一章 误差与范数

数值分析也称计算方法,它不仅是一种研究并解决数学问题的数值近似解的方法,而且是在计算机上使用的解数学问题的方法.它的计算对象是那些在理论上有解,而无法用手工计算的数学问题.

学习数值分析与计算的方法的重要性可以从下面计算机解决实际问题的全过程看出,有了符合实际的数学模型,还必须进行认真的数值分析和选择好的计算方法,才能又快又好地计算出数值结果来.数值分析与计算将提供许多常用的和有价值的计算方法.



数值分析是一门理论性和实践性都很强的课程,它的预备课程是高等数学、线性代数和一门计算机语言.关于计算机语言本书采用计算机软件 MATLAB 6.X 或 7.X.有关该软件的基本操作和编程问题请参考本书所附的光盘中的内容,并且在本书的各章节中也有详细的介绍,使读者在学习数值分析的理论和方法的同时,也学会了该软件的使用.

1.1 误差的来源

1.1.1 误差分析的重要性

在运用数值分析(计算方法)解决实际问题的过程中,会出现各种各样的误差,必须注重误差分析.否则,一种合理的计算也可能得出错误的结果.

例 1.1.1 用差商 $f'(a)\approx\dfrac{f(a+h)-f(a)}{h}$ 求 $f(x)=\ln x$ 在 $x=3$ 处导数的近似值.

(1) 取 $h=0.1$ 和 $h=0.000\,1$,用手工计算,取五位数字计算;

(2) 取 $h=0.1$,$h=0.000\,1$,$h=0.000\,000\,000\,000\,001$ 和 $h=0.000\,000\,000\,000\,000\,1$ 分别用 MATLAB 软件计算,取十五位数字计算;

(3) 比较以上的运算结果,说明是否 $|h|$ 越小则计算结果越准确.

解 根据导数定义,得$f'(3)=\lim\limits_{h\to 0}\dfrac{f(3+h)-f(3)}{h}$,当$|h|$很小时,可以用差商求$f(x)=\ln x$在$x=3$处导数的近似值

$$f'(3)\approx\frac{f(3+h)-f(3)}{h}=\frac{\ln(3+h)-\ln 3}{h}.$$

从理论上讲,$|h|$越小则计算结果越准确.如果用手工计算,取五位数字计算.当$h=0.1$时,得

$$f'(3)\approx\frac{1.131\ 4-1.098\ 6}{0.1}=0.328\ 0,$$

与导数的精确值$f'(3)=\dfrac{1}{3}=0.333\ 3\cdots$比较,这项计算还是可取的.但是当$h=0.000\ 1$时,得

$$f'(3)\approx\frac{1.098\ 6-1.098\ 6}{0.000\ 1}=0,$$

算出的结果反而毫无价值.

如果应用 MATLAB 软件计算,取十五位数字计算,结果就完全不同了.在 MATLAB 工作窗口输入下面程序

```
>> a = 3;h = 0.1;y = log(a + h) - log(a);yx = y/h
```

运行后得$yx=0.327\ 898\ 228\ 229\ 91$.将此程序中$h=0.1$改为$0.000\ 1$,运行后得$yx=0.333\ 327\ 777\ 903\ 85$,后者比前者好.再取$h=0.000\ 000\ 000\ 000\ 001$,运行后得$yx=0.444\ 089\ 209\ 850\ 06$,不如前者好.取$h=0.000\ 000\ 000\ 000\ 000\ 1$,运行后得$yx=0$,算出的结果反而毫无价值.

例 1.1.2 分别求方程组$\boldsymbol{AX}=\boldsymbol{b}$在下列情况时的解,其中$\boldsymbol{A}=\begin{pmatrix}1&1\\1&1.01\end{pmatrix}$.

(1) $\boldsymbol{b}=\begin{pmatrix}2\\2\end{pmatrix}$; (2) $\boldsymbol{b}=\begin{pmatrix}2\\2.01\end{pmatrix}$.

解 (1) 首先将方程组$\boldsymbol{AX}=\boldsymbol{b}$化为同解方程$\boldsymbol{X}=\boldsymbol{A}^{-1}\boldsymbol{b}$,然后在 MATLAB 工作窗口输入程序

```
>> b = [2,2]';A = [1,1;1,1.01]; X = A\b
```

运行后输出当$\boldsymbol{b}=\begin{pmatrix}2\\2\end{pmatrix}$时,$\boldsymbol{AX}=\boldsymbol{b}$的解为$\boldsymbol{X}=\begin{pmatrix}2\\0\end{pmatrix}$;

(2) 同理可得,当$\boldsymbol{b}=\begin{pmatrix}2\\2.01\end{pmatrix}$时,$\boldsymbol{AX}=\boldsymbol{b}$的解为$\boldsymbol{X}=\begin{pmatrix}1\\1\end{pmatrix}$.

可见$\boldsymbol{b}$的微小变化就引起$\boldsymbol{X}$的很大变化,即$\boldsymbol{X}$对$\boldsymbol{b}$的扰动是敏感的.对于一般的方程组$\boldsymbol{AX}=\boldsymbol{b}$,如果解$\boldsymbol{X}$对$\boldsymbol{b}$或$\boldsymbol{A}$的扰动敏感,就称**方程组是病态的**.

1.1.2 误差的来源

科学计算中所处理的数据和计算结果,通常都是在一定精度范围内的近似数值,它们与实际的精确值之间总存在着误差. 引起误差的因素是多方面的,但主要来源有下面几种.

(一) 模型误差

各种实际问题的数学模型,总是在一定条件下抓住主要因素,忽略次要因素加以抽象得出的结果,它是一种理想化了的数学描述,而与实际问题之间总存在误差,这种误差称为模型误差.

例如,建立自由落体运动方程

$$s = \frac{1}{2}gt^2 \tag{1.1}$$

时,忽略了空气阻力等因素. 如果用 $y = f(t)$ 表示真正的运动规律,那么

$$f(t) - \frac{1}{2}gt^2$$

就是数学模型(1.1)的**模型误差**.

(二) 测量误差

数学模型中的已知的参数大多数是通过测量得到的. 而测量过程受到工具、方法、观察者的主观因素、不可预测的随机干扰等因素影响必然带入误差.

例如,自由落体运动方程(1.1)中,当取重力加速度 $g = 9.8\ \mathrm{m/s^2}$ 时,则 $g - 9.8$ 就是**测量误差**.

(三) 截断误差

把无限的计算过程用有限的计算过程代替,这样产生的误差称为**截断误差**.

例 1.1.3 计算 e 的近似值.

解 泰勒级数

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + \frac{x^n}{n!} + \cdots \quad (-\infty < x < \infty),$$

取 $x = 1$,得

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots + \frac{1}{n!} + \cdots. \tag{1.2}$$

这是一个无限过程,计算机无法求到精确值. 只能在(1.2)取有限项时计算,再估计误差. 如果取有限项

$$s_n(1) = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots + \frac{1}{n!}$$

作为 e 的值必然会有误差,根据泰勒余项定理可知其截断误差为

$$e - s_n(1) = \frac{e^{\theta}}{(n+1)!} \quad (0<\theta<1).$$

如果取(1.2)的前九项,输入程序

```
>> n =8;s =1;S =1;
for k =1:n,
s =s * k;S =S +1 /s,
end,
s, S,R =3 /(s * (n +1))
```

或

```
>> S1 =1 +1 +1 /2 +1 /(1 * 2 * 3) +1 /(1 * 2 * 3 * 4) +1 /(1 * 2 * 3 * 4 * 5)
+1 /(1 * 2 * 3 * 4 * 5 * 6) +1 /(1 * 2 * 3 * 4 * 5 * 6 * 7) +1 /(1 * 2 * 3 * 4 * 5 * 6 * 7 *
8),
R1 =3 /(1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9)
```

运行后结果

```
S =                               R =
  2.718 278 769 841 27              8.267 195 767 195 768e -006
```

因为截断误差为

$$e - s_8(1) = \frac{e^{\theta}}{(8+1)!} < \frac{3}{9!} \approx 8.267\,196 \times 10^{-6} \quad (0<\theta<1),$$

所以 e 的近似值 $e \approx s_8(1) = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \frac{1}{6!} + \frac{1}{7!} + \frac{1}{8!} \approx 2.718\,28.$

(四) 舍入误差

在使用计算机或计算器进行数值计算时,每个数据都是用有限位的数字表示的,把一个数表示成具有一定位数的近似数也可能引起误差,这种误差称为**舍入误差**.

例如,本节例 1.1.1 中,在计算 $f'(3)$ 的近似值时,取五位数字用手工计算时,得 $f'(3)\approx 0$ 算出的结果与精确值 $f'(3)=1/3=0.333\,3\cdots$ 相差很大. 取十五位数字应用 MATLAB 软件计算,当 $h=0.000\,1$ 时,运行后得 $f'(3)\approx 0.333\,327\,777\,903\,85$,后者比前者好. 再取 $h=0.000\,000\,000\,000\,001$,运行后得 $f'(3)\approx 0.444\,089\,209\,850\,06$,不如前者好. 取 $h=0.000\,000\,000\,000\,000\,1$,运行后得 $f'(3)\approx 0$,算出的结果反而毫无价值. 这是由舍入误差造成的. 由于计算机硬件只支持有限位机器数的运算,因此在计算中可能引入和传播舍入误差.

在数值分析课程中不讨论和分析模型误差,主要讨论对象是截断误差,它往往是计算中误差的主要部分. 在讲到各种算法时,通过数学方法可以推导出截断误差限的公式. 舍入误差的产生往往带有很大的随机性,讨论比较困难. 在问题本身呈现病态或算法稳定性不好时,它可能成为计算中误差的主要部分. 至于测

量误差,我们把它作为初始的舍入误差看待.

误差分析是一门比较艰难和深奥的学科. 在本课程中主要讨论截断误差和舍入误差. 但是一位训练有素的计算工作者,当发现计算结果与实际不符时,应当能诊断出误差来源,并采取相应的措施加以改进,直至建议对模型的修改.



习 题 1.1

1. 分别求方程组 $AX=b$ 在下列情况时的解,其中 $A=\begin{pmatrix}2 & 2\\ 2 & 2.01\end{pmatrix}$.

(1) $b=\begin{pmatrix}2\\2\end{pmatrix}$;(2) $b=\begin{pmatrix}2\\2.01\end{pmatrix}$.

2. 用差商 $f'(a)\approx\dfrac{f(a+h)-f(a)}{h}$,求 $f(x)=\ln x$ 在 $x=5$ 处导数 $f'(5)$ 的近似值,取十五位数字计算,h 分别取 0.000 1 和 0.000 000 01.

1.2 误差和有效数字

1.2.1 绝对误差和绝对误差限

定义 1.1 设 x 代表准确值 x^* 的一个近似值,称 $e^*=x^*-x$ 为近似值 x 的**绝对误差**,简称**误差**.

误差一般无法准确计算,只能根据测量或计算情况估计出它的绝对值 $|e^*|$ 的一个上限. 设存在一个正数 ε,使

$$|e^*|=|x^*-x|\leqslant\varepsilon, \tag{1.3}$$

称 ε 为近似值 x 的**绝对误差限**,实际把它简称为**误差**、**误差限**、**绝对误差**或**精度**.

由(1.3)式可得

$$x-\varepsilon\leqslant x^*\leqslant x+\varepsilon,$$

在工程中常记为

$$x^*=x\pm\varepsilon.$$

例如,$l=(20.1\pm0.05)$ mm. 再如,用毫米刻度的米尺测量不超过 1 m 的长度 x^*,读法如下:如果长度接近于毫米刻度 x,就读出那个刻度数 x 作为该长度 x^* 的近似值. 显然这个近似数 x 的绝对误差为 0.05 mm. 由这两个例子可以看出绝对误差 e^* 是有量纲单位的. $|e^*|$ 的大小标志着近似值 x 的精度. 一般的,在同一量纲的不同近似值中,$|e^*|$ 越小,x 的精度就越高.

1.2.2 相对误差和相对误差限

绝对误差的大小不能完全刻画近似值的准确程度. 例如,测量百米跑道产生 10 cm 的误差与测量航天飞机上的一件长为 1 cm 零件的长度产生 0.01 cm 的误差,我们不能简单地认为后者更精确,还应该考虑被测量值的大小,这就有必要引入相对误差的概念.

定义 1.2 绝对误差 e^* 与准确值 x^* 的比值

$$e_r^* = \frac{e^*}{x^*} = \frac{x^* - x}{x^*}$$

称为 x^* 的近似值 x 的**相对误差**.

相对误差是无量纲的量,常用百分比表示. 相对误差反映了一个近似数的准确程度,$|e_r^*|$越小,x 的精度就越高. 但由于准确值 x^* 总是不知道的,因此,在实际问题中,常取相对误差为

$$e_r^* = \frac{e^*}{x} = \frac{x^* - x}{x}.$$

上例中,百米跑道产生 10 cm 的绝对误差所对应的相对误差为 0.001,长为 1 cm 的零件的长度产生 0.01 cm 的绝对误差所对应的相对误差为 0.01. 后者绝对误差小,前者相对误差小,前者精度比后者高.

相对误差一般也无法准确计算,只能根据测量或计算情况估计出它的绝对值$|e_r^*|$的一个上限. 设存在一个正数 ε_r,使

$$|e_r^*| = \left|\frac{e^*}{x}\right| = \left|\frac{x^* - x}{x}\right| \leqslant \frac{\varepsilon}{|x|} = \varepsilon_r \tag{1.4}$$

称 ε_r 为近似值 x 的**相对误差限**,实际把它作为**相对误差**.

由(1.4)式可知,相对误差限与绝对误差限有如下关系

$$\varepsilon = |x|\varepsilon_r.$$

例 1.2.1 取 2.718 28 作为 e 的四舍五入近似值时,求其绝对误差和相对误差.

解 在 MATLAB 工作窗口输入程序

```
>> juewu = exp(1) - 2.71828
```

因为运行后输出结果为 juewu = 1.828 459 045 505 326e 006,所以,2.718 28的绝对误差为 $\varepsilon = 0.182\ 846 \times 10^{-5}$,相对误差为

$$\varepsilon_r = \frac{\varepsilon}{|x|} = \frac{0.182\ 85 \times 10^{-5}}{2.718\ 28} < 6.726\ 54 \times 10^{-7} < 0.000\ 07\%.$$

例 1.2.2 计算 $\int_0^{\frac{\pi}{2}} \frac{\sin x}{x} \mathrm{d}x$ 的近似值,并确定其绝对误差和相对误差.

解 因为被积函数$\frac{\sin x}{x}$的原函数不是初等函数，故用泰勒级数求之.

$$\frac{\sin x}{x}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\frac{x^8}{9!}+\cdots \quad (-\infty<x<\infty), \tag{1.5}$$

这是一个无限过程，计算机无法求到精确值. 可用(1.5)的前四项 $1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}$代替被积函数$\frac{\sin x}{x}$，得

$$y=\int_0^{\frac{\pi}{2}}\frac{\sin x}{x}\mathrm{d}x\approx\int_0^{\frac{\pi}{2}}\left(1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}\right)\mathrm{d}x=\frac{\pi}{2}-\frac{\left(\frac{\pi}{2}\right)^3}{3\cdot 3!}+\frac{\left(\frac{\pi}{2}\right)^5}{5\cdot 5!}-\frac{\left(\frac{\pi}{2}\right)^7}{7\cdot 7!}=\hat{y}.$$

根据泰勒余项定理和交错级数收敛性的判别定理，得到绝对误差

$$R=|y-\hat{y}|<\frac{\left(\frac{\pi}{2}\right)^9}{9\cdot 9!}=WU,$$

在 MATLAB 命令窗口输入计算程序如下：

```
syms x
f =1 -x^2/(1*2*3) +x^4/(1*2*3*4*5) -x^6/(1*2*3*4*5*
6*7)
y =int(f,x,0,pi/2),y1 =double(y)
y11 =pi/2 -(pi/2)^3/(3*3*2) +(pi/2)^5/(5*5*4*3*2) -
(pi/2)^7/(7*7*6*5*4*3*2)
inf =int(sin(x)/x,x,0,pi/2),infd =double(inf),
WU =(pi/2)^9/(9*9*8*7*6*5*4*3*2), R =infd -y11
```

因为运行后输出结果为：$y=1.370\,762\,168\,154\,49$，$\hat{y}=1.370\,744\,664\,189\,38$，$R=1.750\,396\,510\,491\,47\mathrm{e}-005$，$WU=1.782\,679\,830\,970\,664\mathrm{e}-005<10^{-4}$. 所以，$\hat{y}$ 的绝对误差为 $\varepsilon=10^{-4}$，故 $y=\int_0^{\frac{\pi}{2}}\frac{\sin x}{x}\mathrm{d}x\approx 1.370\,7$. $\hat{y}$ 的相对误差为

$$\varepsilon_r=\frac{\varepsilon}{|\hat{y}|}=\frac{10^{-4}}{1.370\,7}<0.007\,3\%.$$

1.2.3 有效数字

为了便于了解 x 作为准确值 x^* 的一个近似值具有几位有效数字的取值范围，我们用相对误差引入下面定义.

(一) 有效数字的定义

定义 1.3 设数 x^* 的近似值 $x=\pm 0.x_1x_2\cdots x_n\times 10^m$，其中 $x_i(i=1,2,\cdots,n)$ 是 0 到 9 之间的任意一个非负整数，且 $x_1\neq 0$，n 是正整数，m 是整数. 如果绝对

误差 e^* 的

$$|e^*| = |x^* - x| \leqslant \frac{1}{2} \times 10^{m-l}, 1 \leqslant l \leqslant n \tag{1.6}$$

则称 x 为 x^* 的具有 l **位有效数字**的近似值，x 准确到第 l 位，$x_1x_2\cdots x_l$ 为 x 的**有效数字**.

例 1.2.3 试确定 1.370 7 为 $y = \int_0^{\frac{\pi}{2}} \frac{\sin x}{x}\mathrm{d}x$ 的具有几位有效数字的近似值.

解 由例 1.2.2 可知，

$$|y - 1.370\ 7| < 0.000\ 062\ 168 < \frac{1}{2} \times 10^{-3},$$

因为 $m - l = -3$，$m = 1$，所以，$l = 4$. 因此，1.370 7 为 y 的具有 4 位有效数字的近似值.

例 1.2.4 取 2.718 28 作为 e 的四舍五入近似值时，具有几位有效数字.

解 由例 1.2.1 可知，

$$|\mathrm{e} - 2.718\ 28| < 0.000\ 001\ 828\ 5 < 0.5 \times 10^{-5},$$

因为 $m - l = -5$，$m = 1$，所以，$l = 6$. 因此，2.718 28 为 e 的具有 6 位有效数字的近似值.

（二）关于有效数字的几点说明

（1）$x = \pm 0.x_1x_2\cdots x_n \times 10^m$ 作为数 x^* 的近似值时，$x_1x_2\cdots x_n$ 不一定为 x 的有效数字. 但是用四舍五入取准确值 x^* 的前 l 位作为近似值 $x = 0.x_1x_2\cdots x_l \times 10^m$，则 x 必有 l 个有效数字 $x_1x_2\cdots x_l$.

例如，在例 1.2.4 中取 2.718 28 作为 e 的四舍五入近似值时，2.718 28 具有 6 位有效数字.

再如，取 2.718 281 作为 e 的近似值时，由于

$$|\mathrm{e} - 2.718\ 281| < 0.828\ 459 \times 10^{-6} < 0.5 \times 10^{-5},$$

因此，2.718 281 为 e 的具有 6 位有效数字近似值；其有效数字为 271 828.

（2）把任何数 $x = 0.x_1x_2\cdots x_n \times 10^m$ 乘以 10^p（$p = \pm 1, \pm 2, \cdots$）等于移动该数的小数点后得 $x = 0.x_1x_2\cdots x_n \times 10^{m+p}$，这并不影响 x 的有效数字.

例如，2.718 28 和 2 718.28 都具有 6 位有效数字. 但 2 718.280 和 2 718.28 的有效数字的位数不同，前者有 7 位，后者有 6 位有效数字. 因此，要注意诸如 0.2，0.20，0.200，…的不同含义，有效数字零不能丢掉.

（3）有效数字相同的两个近似数的绝对误差不一定相同.

例如，2.718 28 和 2 718.28 都具有 6 位有效数字；但绝对误差不一定相同.

（4）准确值被认为具有无穷多位有效数字.

例如,直角三角形面积 $S=\frac{1}{2}xy=0.5xy$,不能认为公式中 $\frac{1}{2}$ 用 0.5 表示时,0.5 只有一位有效数字.因为 0.5 是准确值,没有误差,即 $\varepsilon=0$,对于任意的正整数 n(包括 $n\to\infty$),(1.6)式恒成立,由定义 1.3 知,准确值 0.5 具有无穷多位有效数字.

1.2.4 有效数字与误差的关系

从有效数字的定义 1.3 可以看出有效数字与绝对误差的关系,有效数字的位数越多,对应的绝对误差就越小.下面的定理对此进行了具体的讨论.

定理 1.1 设数 x^* 的近似值 $x=\pm 0.x_1x_2\cdots x_n\times 10^m$ 具有 n 位有效数字,则 x 的相对误差满足下列不等式

$$|e_r^*|\leqslant\frac{1}{2x_1}\times 10^{1-n}. \tag{1.7}$$

证明 因为数 x^* 的近似值 $x=\pm 0.x_1x_2\cdots x_n\times 10^m$ 具有 n 位有效数字,由定义 1.3 得

$$|e^*|=|x^*-x|\leqslant\frac{1}{2}\times 10^{m-n},\text{且}|x|\geqslant x_1\times 10^{m-1},$$

所以
$$|e_r^*|=\left|\frac{x^*-x}{x}\right|\leqslant\frac{\frac{1}{2}\times 10^{m-n}}{x_1\times 10^{m-1}}=\frac{1}{2x_1}\times 10^{1-n},$$

因此,(1.7)式成立.

定理中的条件是充分条件,而不是必要条件.近似数的有效数字越多,其相对误差就越小.但是,相对误差越小,有效数字的位数只是可能多.

一般应用中,可以取相对误差限

$$\varepsilon_r=\frac{1}{2x_1}\times 10^{1-n}, \tag{1.8}$$

由于 n 越大,ε_r 越小,所以,有效数字位数越多,相对误差就越小.

例 1.2.5 取 3.141 592 653 589 8 作为 π 的四舍五入近似值时,试求其相对误差.

解 四舍五入的近似值 3.141 592 653 589 8 的各位都是有效数字,即具有 $n=14$ 位有效数字,且 $x_1=3$,代入(1.7)式,得

$$|e_r^*|\leqslant\frac{1}{2\times 3}\times 10^{1-14}<0.166\,666\,666\,666\,7\times 10^{-13}$$

例 1.2.6 设任意一个实数 x^* 的近似值 $x=-0.x_1x_2\cdots x_n\times 10^m$ 具有 n 位有效数字,估计 x 的相对误差 ε_r 的最大值和最小值.

解 设与(1.8)式中的 ε_r 对应的连续函数为 $y=\frac{10^{1-n}}{2t}$ $(1\leqslant t\leqslant 9)$.因为

$\frac{dy}{dt}=-\frac{10^{1-n}}{2t^2}<0$，所以，$y$ 在区间[1,9]上严格单调减少. 由单调减少的定义知，当 $x=1$ 时，相对误差有最大值 $\varepsilon_r=\frac{1}{2}\times10^{1-n}=0.5\times10^{1-n}$；当 $x=9$ 时，相对误差有最小值 $\varepsilon_r=\frac{1}{18}\times10^{1-n}\approx0.56\times10^{-n}$.

定理 1.2 设数 x^* 的近似值 $x=\pm0.x_1x_2\cdots x_n\times10^m$ 的相对误差 e_r^* 满足下列不等式

$$|e_r^*|\leqslant\frac{1}{2(x_1+1)}\times10^{1-n},\tag{1.9}$$

则 x 具有 n 位有效数字.

证明 因为 $|x|\leqslant(x_1+1)\times10^{m-1}$. 故按题设(1.9)式，有

$$|x^*-x|=\left|\frac{x^*-x}{x}\right||x|\leqslant\frac{1}{2(x_1+1)}\times10^{1-n}\times(x_1+1)\times10^{m-1}$$
$$=\frac{1}{2}\times10^{m-n},$$

因此，x 具有 n 位有效数字.

例 1.2.7 设 sin 9°的近似值 $x=0.156\ 431$ 的相对误差 $e_r^*<0.25\times10^{-4}$，问 x 具有几位有效数字？

解 把 $x_1=1$ 和 $e_r^*<0.25\times10^{-4}$代入式(1.9)，得

$$|e_r^*|\leqslant\frac{1}{2(1+1)}\times10^{1-n}=0.25\times10^{-4},$$

所以，$1-n=-4$，解得 $n=5$. 因此，x 具有 5 位有效数字.

例 1.2.8 表 1－1 中各 x^* 都是对准确值 x 进行四舍五入得到的近似值. 试分别指出其绝对误差限、相对误差限及有效数字位数.

表 1－1

x^*	绝对误差限	相对误差限	有效数字位数
0.301 2			
30.12			
30.120			
30 120			
$0.301\ 2\times10^5$			

解 (1) 因为用四舍五入取准确值 x^* 的前 n 位作为近似值 $x=0.x_1x_2\cdots x_n\times10^m$，则 x 必有 n 个有效数字 $x_1x_2\cdots x_n$. 且各数 0.301 2，$30.12=0.301\ 2\times10^2$，

$0.301\ 2\times10^5$ 都是对准确值 x^* 进行四舍五入得到的近似值，所以 0.301 2，30.12，$0.301\ 2\times10^5$ 都有 4 位有效数字 3 012. 而 $30.120=0.301\ 20\times10^2$ 和 $30\ 120=0.301\ 20\times10^5$ 都有 5 位有效数字 30 120.

说明 把任何数 $x=0.x_1x_2\cdots x_n\times10^m$ 乘以 $10^p(p=\pm1,\pm2,\cdots)$ 等于移动该数的小数点得 $x=0.x_1x_2\cdots x_n\times10^{m+p}$，这并不影响 x 的有效数字.

（2）根据有效数字的定义可得，具有四位有效数字的数 0.301 2，$30.12=0.301\ 2\times10^2$，$0.301\ 2\times10^5$ 的绝对误差限分别为 $|x^*-0.301\ 2|\leqslant\frac{1}{2}\times10^{0-4}=0.5\times10^{-4}$，$|x^*-30.12|\leqslant\frac{1}{2}\times10^{2-4}=0.5\times10^{-2}$，$|x^*-0.301\ 2\times10^5|\leqslant\frac{1}{2}\times10^{5-4}=5$.

具有五位有效数字的数 $30.120=0.301\ 20\times10^2$ 和 $30\ 120=0.301\ 20\times10^5$ 的绝对误差限分别为 $|x^*-30.120|\leqslant\frac{1}{2}\times10^{2-5}=0.5\times10^{-3}$，$|x^*-30\ 120|\leqslant\frac{1}{2}\times10^{5-5}=0.5$.

（3）根据定理 1.1 可知，具有四位有效数字的数 0.301 2，$30.12=0.301\ 2\times10^2$，$0.301\ 2\times10^5$ 的相对误差限都为

$$|e_r^*|\leqslant\frac{1}{2x_1}\times10^{1-n}=\frac{1}{2\times3}\times10^{1-4}=\frac{1}{6}\times10^{-3}.$$

而具有五位有效数字的数 $30.120=0.301\ 20\times10^2$ 和 $30\ 120=0.301\ 20\times10^5$ 的相对误差限都为

$$|e_r^*|\leqslant\frac{1}{2x_1}\times10^{1-n}=\frac{1}{2\times3}\times10^{1-5}=\frac{1}{6}\times10^{-4}.$$

表 1-2

x^*	绝对误差限	相对误差限	有效数字位数
0.301 2	0.5×10^{-4}	$\frac{1}{6}\times10^{-3}$	4 位有效数字
30.12	0.5×10^{-2}	$\frac{1}{6}\times10^{-3}$	4 位有效数字
30.120	0.5×10^{-3}	$\frac{1}{6}\times10^{-4}$	5 位有效数字
30 120	0.5	$\frac{1}{6}\times10^{-4}$	5 位有效数字
$0.301\ 2\times10^5$	5	$\frac{1}{6}\times10^{-3}$	4 位有效数字



习 题 1.2

1. 为了使 $\sqrt{20}=4.4\cdots$ 的近似值的相对误差小于0.1%，试问取几位有效数字？
2. 写出光速 $c^*=(2.997\,925\pm0.000\,001)\times10^{10}$ cm/s 的绝对误差和相对误差.
3. 取3.141 592 65作为 π 的四舍五入近似值时，求其有效数字、绝对误差和相对误差.
4. 问3.142，3.141，$\frac{22}{7}$ 分别作为 π 的近似值各具有几位有效数字？

1.3 误差估计的基本方法

1.3.1 误差估计的运算

数据运算中的误差的估计情况比较复杂，通常利用泰勒展开式的方法估计误差.

(一) 误差估计的一般运算

定理 1.3 设所有的初始数据 $x_1^*,x_2^*,\cdots,x_n^*$ 是相互独立的自变量，而通过某个算式所得到的结果看做是这些初始数据的函数，表示为

$$Z^*=f(x_1^*,x_2^*,\cdots,x_n^*),\tag{1.10}$$

$x_1,x_2,\cdots,x_n$ 是 $x_1^*,x_2^*,\cdots,x_n^*$ 的近似值，且每个 x_i 的绝对误差 $e^*(x_i)=x_i^*-x_i$ 的绝对值都很小，多元函数 f 在点 $x=(x_1,x_2,\cdots,x_n)$ 处可微，则 $Z=f(x_1,x_2,\cdots,x_n)$ 的绝对误差为

$$e^*(Z)=Z^*-Z\approx\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)_x e^*(x_i),\tag{1.11}$$

相对误差为

$$e_r^*(Z)\approx\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)_x\frac{x_i\cdot e_r^*(x_i)}{Z}.\tag{1.12}$$

证明 因为对于 Z^* 的近似值

$$Z=f(x_1,x_2,\cdots,x_n)\tag{1.13}$$

的绝对误差可以利用泰勒级数展开得

$$\begin{aligned}e^*(Z)&=Z^*-Z=f(x_1^*,x_2^*,\cdots,x_n^*)-f(x_1,x_2,\cdots,x_n)\\&\approx\left(\frac{\partial f}{\partial x_1}\right)_x(x_1^*-x_1)+\left(\frac{\partial f}{\partial x_2}\right)_x(x_2^*-x_2)+\cdots+\left(\frac{\partial f}{\partial x_n}\right)_x(x_n^*-x_n)\\&=\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)_x(x_i^*-x_i)=\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)_x e^*(x_i),\end{aligned}\tag{1.14}$$

而 x_i 的相对误差

$$e_r^*(x_i)=\frac{e^*(x_i)}{x_i}=\frac{x_i^*-x_i}{x_i}\quad(i=1,2,\cdots,n).\tag{1.15}$$

由(1.14)式和(1.15)式,可得到 Z 的相对误差为

$$e_r^*(Z)=\frac{e^*(Z)}{Z}\approx\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)_x\frac{(x_i^*-x_i)}{Z}=\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)_x\frac{x_i\cdot e_r^*(x_i)}{Z}.$$

(二) 误差估计的四则运算

我们可以利用(1.11)式和(1.12)式来估计按函数 f 的计算误差,给定函数 f 的具体形式,就可以得出函数的具体的估计. 特别可以得到和、差、积、商的误差估计.

推论 1.1 设初始数据 x^*,y^* 是相互独立的自变量,x,y 分别是 x^*,y^* 的近似值,它们的绝对误差分别为 $e^*(x)=x^*-x,e^*(y)=y^*-y$,且其绝对值都很小,则进行加、减、乘、除运算时,初始数据的误差估计和计算结果中产生的误差之间有下面关系:

(1) 若 $f(x^*,y^*)=x^*\pm y^*$,则

$$e^*(x\pm y)\approx e^*(x)\pm e^*(y);\tag{1.16}$$

$$e_r^*(x\pm y)\approx\frac{x}{x\pm y}e_r^*(x)\pm\frac{y}{x\pm y}e_r^*(y);\tag{1.17}$$

$$|e^*(x\pm y)|\leqslant|e^*(x)|+|e^*(y)|;\tag{1.18}$$

$$|e_r^*(x\pm y)|\leqslant\left|\frac{x}{x\pm y}\right||e_r^*(x)|+\left|\frac{y}{x\pm y}\right||e_r^*(y)|.\tag{1.19}$$

(2) 若 $f(x^*,y^*)=x^*\cdot y^*$,则

$$e^*(x\cdot y)\approx ye^*(x)+xe^*(y);\tag{1.20}$$

$$e_r^*(x\cdot y)\approx e_r^*(x)+e_r^*(y);\tag{1.21}$$

$$|e^*(x\cdot y)|\leqslant|y||e^*(x)|+|x||e^*(y)|;\tag{1.22}$$

$$|e_r^*(x\cdot y)|\leqslant|e_r^*(x)|+|e_r^*(y)|.\tag{1.23}$$

(3) 若 $f(x^*,y^*)=\dfrac{x^*}{y^*}$,则

$$e^*\left(\frac{x}{y}\right)\approx\frac{1}{y}e^*(x)-\frac{x}{y^2}e^*(y);\tag{1.24}$$

$$e_r^*\left(\frac{x}{y}\right)\approx e_r^*(x)-e_r^*(y);\tag{1.25}$$

$$\left|e^*\left(\frac{x}{y}\right)\right|\leqslant\left|\frac{1}{y}\right||e^*(x)|+\left|\frac{x}{y^2}\right||e^*(y)|;\tag{1.26}$$

$$\left|e_r^*\left(\frac{x}{y}\right)\right|\leqslant|e_r^*(x)|+|e_r^*(y)|.\tag{1.27}$$

证明 (1) 设 $f(x,y)=x-y$，则 $\frac{\partial f}{\partial x}=1,\frac{\partial f}{\partial y}=-1$，代入(1.11)式，得 $f(x,y)=x-y$ 的绝对误差为

$$e^*(x-y)\approx\left(\frac{\partial f}{\partial x}\right)_{(x,y)}e^*(x)+\left(\frac{\partial f}{\partial y}\right)_{(x,y)}e^*(y)=e^*(x)-e^*(y).$$

由(1.12)式得 $f(x,y)=x-y$ 的相对误差为

$$e_r^*(x-y)\approx\left(\frac{\partial f}{\partial x}\right)_{(x,y)}\frac{x\cdot e_r^*(x)}{x-y}+\left(\frac{\partial f}{\partial y}\right)_{(x,y)}\frac{y\cdot e_r^*(y)}{x-y}=\frac{x}{x-y}e_r^*(x)-\frac{y}{x-y}e_r^*(y).$$

(3) 设 $f(x,y)=\frac{x}{y}$，则 $\frac{\partial f}{\partial x}=\frac{1}{y},\frac{\partial f}{\partial y}=-\frac{x}{y^2}$，代入(1.11)式，得 $f(x,y)=\frac{x}{y}$ 的绝对误差为

$$e^*\left(\frac{x}{y}\right)\approx\left(\frac{\partial f}{\partial x}\right)_{(x,y)}e^*(x)+\left(\frac{\partial f}{\partial y}\right)_{(x,y)}e^*(y)=\frac{1}{y}e^*(x)-\frac{x}{y^2}e^*(y).$$

由(1.12)式得 $f(x,y)=\frac{x}{y}$ 的相对误差为

$$e_r^*\left(\frac{x}{y}\right)\approx\left(\frac{\partial f}{\partial x}\right)_{(x,y)}\frac{x\cdot e_r^*(x)}{\frac{x}{y}}+\left(\frac{\partial f}{\partial y}\right)_{(x,y)}\frac{y\cdot e_r^*(y)}{\frac{x}{y}}=e_r^*(x)-e_r^*(y).$$

请读者完成其余的证明.

例 1.3.1 已测得某场地长 x^* 的值 $x=210$ m，宽 y^* 的值 $y=180$ m，已知 x 的绝对误差是0.3 m，y 的绝对误差是0.2 m，求场地面积 $S=xy$ 的绝对误差和相对误差.

解 由(1.22)式和(1.23)式，得面积 $S=xy$ 的绝对误差

$$|e^*(S)|\leqslant|y||e^*(x)|+|x||e^*(y)|=180\times0.3+210\times0.2=96\ (\mathrm{m}^2)$$

和相对误差

$$|e_r^*(S)|\leqslant|e_r^*(x)|+|e_r^*(y)|=\frac{0.3}{210}+\frac{0.2}{180}\approx0.25\%.$$

例 1.3.2 设 $y=a^x(a>0,a\neq1)$，求 y 的相对误差与 x 的相对误差的关系式.

解 因为 y 的绝对误差与 x 的绝对误差的关系式为

$$e^*(y)\approx\mathrm{d}\,y=\mathrm{d}\,a^x=a^xe^*(x)\ln a,$$

所以，y 的相对误差与 x 的相对误差的关系式为

$$e_r^*(y)=\frac{e^*(y)}{y}\approx\frac{\mathrm{d}y}{y}=\frac{a^x\ln a\ e^*(x)}{a^x}=x\frac{e^*(x)}{x}\ln a=xe_r^*(x)\ln a.$$

例 1.3.3 若用电表测得一个电阻两端的电压和流过的电流分别为 $V=(110\pm2)$ V，$I=(20\pm0.5)$ A，试由欧姆定律 $R=\frac{V}{I}$ 求这个电阻值 R 的近似值，并

求所得近似值的绝对误差限与相对误差限.

解 根据 $V=(110\pm2)\,\mathrm{V}, I=(20\pm0.5)\,\mathrm{A}$ 知,这个电阻两端的电压和流过的电流的近似值分别为 $V\approx110\ \mathrm{V}, I\approx20\ \mathrm{A}$,它们的绝对误差限分别为 $e^*(V)=V^*-V=2\ \mathrm{V}, e^*(I)=I^*-I=0.5\ \mathrm{A}$. 从而相对误差限分别为

$$e_r^*(V)=\frac{e^*(V)}{V}=\frac{2}{110}\approx1.82\%, e_r^*(I)=\frac{e^*(I)}{I}=\frac{0.5}{20}=2.5\%.$$

这个电阻值 R 的近似值为

$$R=\frac{V}{I}\approx\frac{110}{20}=5.5(\Omega),$$

其绝对误差限为

$$|e^*(R)|=\left|e^*\left(\frac{V}{I}\right)\right|\leqslant\left|\frac{1}{I}\right||e^*(V)|+\left|\frac{V}{I^2}\right||e^*(I)|$$
$$=\frac{1}{20}\times2+\frac{110}{20^2}\times0.5=23.75\%,$$

相对误差限为

$$|e_r^*(R)|=\left|e_r^*\left(\frac{V}{I}\right)\right|\leqslant|e_r^*(V)|+|e_r^*(I)|=\frac{2}{110}+\frac{0.5}{20}<4.32\%.$$

1.3.2 数值的稳定性

计算一个数学问题,首先需要预先设计好由已知数据计算问题的运算顺序,这就是**算法**. 初始数据的误差通常通过一系列的计算进行传播. 因为初始条件下的小误差对最终结果产生的影响较小,所以对任何数值计算都要尽量减少初始误差,这样的算法称为**稳定的算法**. 否则,称为**不稳定的算法**. 在数值分析中应当尽量选择稳定的算法.

定义 1.4 设 ε_0 表示初始误差,ε_n 表示第 n 步计算后误差的增长.

(1) 如果 $|\varepsilon_n|\approx n\varepsilon_0$,则称误差按线性增长;

(2) 如果 $|\varepsilon_n|\approx c^n\varepsilon_0$,则称误差按指数增长;

① 如果 $c>1$,则当 $n\to\infty$ 时,误差按指数无界增长;

② 如果 $0<c<1$,则当 $n\to\infty$ 时,误差按指数的增长趋于零.

下面的例子揭示了初始误差的稳定性传播和不稳定性传播.

例 1.3.4 设计三种算法求方程 $2x^2+x-15=0$ 在 $(2,3)$ 的一个正根 x^* 的近似值,并研究每种算法的误差传播情况.

解 为解已知方程,我们可以设计如下三种算法,然后将计算结果与此方程的精确解 $x^*=2.5$ 比较,观察误差的传播.

算法 1 将已知方程化为同解方程 $x=15-2x^2$. 取初值 $x_0=2$,按迭代公式

$$x_{k+1}=15-2x_k^2$$

依次计算 $x_1,x_2,\cdots,x_n,\cdots$,结果列入表 1－3 中.

算法 2 将已知方程化为同解方程 $x=\frac{15}{2x+1}$. 取初值 $x_0=2$,按迭代公式

$$x_{k+1}=\frac{15}{2x_k+1}$$

依次计算 $x_1,x_2,\cdots,x_n,\cdots$,结果列入表 1－3 中.

算法 3 将已知方程化为同解方程 $x=x-\frac{2x^2+x-15}{4x+1}$. 取初值 $x_0=2$,按迭代公式为

$$x_{k+1}=x_k-\frac{2x_k^2+x_k-15}{4x_k+1}$$

依次计算 $x_1,x_2,\cdots,x_n,\cdots$,结果列入表 1－3 中.

我们为这三种算法的计算编写两套 MATLAB 程序如下:

(1) MATLAB 主程序

```
function [k,juecha,xiangcha,xk] = liti112(x0,x1,limax)
% 输入的量 - -x0 是初值,limax 是迭代次数和精确值 x1;
% 输出的量 - -每次迭代次数 k 和迭代值 xk,
%          - -每次迭代的绝对误差 juecha 和相对误差 xiangcha,
x(1) -x0;
for i =1:limax
    x(i +1) - f1(x(i));% 程序中调用的 f1.m
    juecha = abs(x(i) -x1);
    xiangcha = juecha /( abs(x(i)) +eps);
    xk =x(i);k =i -1;[(i -1),juecha,xiangcha,xk]
end
```

(2) MATLAB 调用函数程序及其计算结果

① 算法 2 的 MATLAB 调用函数程序

```
function y1 = f1(x)
            y1 =15 /(2 * x +1);
```

② 将 MATLAB 主程序和调用函数程序分别命名 liti112.m 和 f1.m,分别保存为 M 文件,然后在 MATLAB 工作窗口输入命令

```
>> [k,juecha,xiangcha,xk] = liti112(2,2.5,100)
```

③ 运行后输出计算结果列入表 1－3 和表 1－4 中.

④ 将算法 2 的 MATLAB 调用函数程序的函数分别用 `y1 =15 -2 * x^2` 和 `y1 =x -(2 * x^2 +x -15) /(4 * x +1)` 代替,得到算法 1 和算法 3 的调用函数程序,将其保存,运行后将三种算法的前 8 个迭代值 $x_1,x_2,\cdots,x_8$ 列在一起(见

表 1－3)，进行比较. 将三种算法的 $x_1, x_2, \cdots, x_8$ 对应的绝对误差和相对误差的值列在一起(见表 1－4)，进行比较.

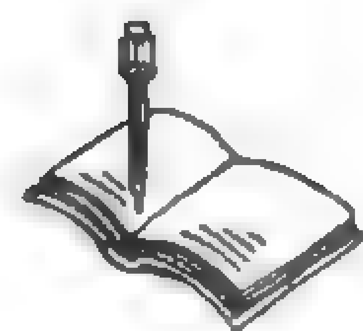
表 1－3 例 1.3.4 中三种算法的计算结果

迭代次数 \ 算法	算法 1 的迭代结果	算法 2 的迭代结果	算法 3 的迭代结果
0	2	2.000 000 00	2.000 000 00
1	7	3.000 000 00	2.555 555 56
2	－83	2.142 857 14	2.500 550 06
3	－13 763	2.837 837 84	2.500 000 06
4	－378 840 323	2.246 963 56	2.500 000 00
5	$-2.870\,4\times10^{17}$	2.246 963 56	2.500 000 00
6	$-1.647\,8\times10^{35}$	2.321 774 84	2.500 000 00
7	$-5.430\,7\times10^{70}$	2.657 901 65	2.500 000 00
⋮	⋮	⋮	⋮
99	－Inf	2.500 000 01	2.500 000 00

由表 1－3 和表 1－4 可以看出，算法 1 的初始绝对误差和相对误差分别是 4.500 000 00 和 0.642 857 14，它们是三种算法中最大者，由算法 1 得到的迭代序列 $\{x_k\}$ 对应的绝对误差是不稳定的，且当 $n\to\infty$ 时，按指数无界增长，相对误差趋近于不定值，$x_k\to-\infty$，即 $\{x_k\}$ 不收敛；算法 2 的初始绝对误差和相对误差分别是 0.500 000 00 和 0.166 666 67，它们是三种算法中居于中间，由算法 2 得到的迭代序列 $\{x_k\}$ 对应的绝对误差和相对误差是稳定的，且当 $n\to\infty$ 时，按指数缓慢递减且趋近于 0，$\{x_k\}$ 缓慢地摆动式收敛于精确解 2.5(参见算法 2 的运行结果)；算法 3 的初始绝对误差和相对误差分别是 0.055 555 60 和 0.021 739 13，它们是三种算法中最小者，由算法 3 得到的迭代序列 $\{x_k\}$ 对应的绝对误差和相对误差是稳定的，且当 $n\to\infty$ 时，按指数快速递减且趋近于 0，$\{x_k\}$ 快速地递减且收敛于精确解 2.5.

表 1-4 例 1.3.4 中三种算法计算结果的误差

算法 迭代次数	算法 1 的误差		算法 2 的误差		算法 3 的误差	
	绝对误差	相对误差	绝对误差	相对误差	绝对误差	相对误差
0	0.500 000 00	0.250 000 00	0.500 000 00	0.250 000 00	0.500 000 00	0.250 000 00
1	4.500 000 00	0.642 857 14	0.500 000 00	0.166 666 67	0.055 555 60	0.021 739 13
2	85.500 000 00	1.030 120 48	0.357 142 86	0.166 666 670	0.000 550 10	0.000 219 97
3	13 765.500 00	0.000 100 02	0.337 837 84	0.119 047 62	0.000 000 06	0.000 000 02
4	378 840 326	1.000 000 01	0.253 036 44	0.112 612 62	0.000 000 00	0.000 000 00
5	$2.870\ 399\ 81\times10^{17}$	1	0.230 287 04	0.084 345 48	0	0
6	$1.647\ 839\ 01\times10^{35}$	1	0.178 225 16	0.076 762 47	0	0
7	$5.430\ 746\ 80\times10^{70}$	1	0.157 901 65	0.059 408 39	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
99	Inf	NaN	0.000 000 01	0.000 000 00	0	0



习 题 1.3

1. 测得电压 $V=(110\pm2)$ V，电流 $I=(20\pm0.5)$ A，则由欧姆定律得 $R=\dfrac{V}{I}=5.5\ \Omega$，求 R 的绝对误差和相对误差.

2. 设 $y=x^n$，求 y 的相对误差与 x 的相对误差的关系式.

3. 设计三种算法求方程 $2x^2+x-3=0$ 在 $(1,2)$ 的一个正根 x^* 的近似值，并研究每种算法的误差传播情况.

4. 已测得某场地长 x^* 的值 $x=110$ m，宽 y^* 的值 $y=80$ m，已知 x 的绝对误差是 0.2 m，y 的绝对误差是 0.1 m，求场地面积 $S=xy$ 的绝对误差和相对误差.

5. 正方形的边长约为 200 cm，怎样测量才能使其面积误差不超过 1 cm^2？

1.4 数值计算中应注意的问题

从例 1.3.4 可以看出，一个问题的解决，往往要经过多次运算. 每一步运算都可能产生误差，在反复多次计算的过程中，必然产生误差的传播和积累. 显然，当误差积累偏大时，会使计算结果失真. 因此，在每一步计算中，都应该防止产生

误差升级的现象. 下面通过对误差传播规律的分析,提出一些应该注意的问题.

1.4.1 避免两个相近的数相减

从(1.17)式和(1.19)式知$f(x,y)=x-y$的相对误差为

$$e_r^*(x-y)\approx\frac{x}{x-y}e_r^*(x)-\frac{y}{x-y}e_r^*(y),$$

$$|e_r^*(x-y)|\leqslant\left|\frac{x}{x-y}\right||e_r^*(x)|+\left|\frac{y}{x-y}\right||e_r^*(y)|,$$

当x与y的值很相近时,$f(x,y)$的相对误差可能变大,导致有效数位下降. 因此,在数值计算中要避免两个相近的数相减.

避免两个相近的数相减的方法很多. 下面介绍四种常用的方法.

方法1 倒数变换法

如果两个相近的数x与y相减,设$z_1=x-y$,$z_2=x+y$,且$z_1\cdot z_2=1$,那么将z_1化为下面形式计算

$$z_1=\frac{x^2-y^2}{x+y}=\frac{1}{x+y}.$$

如果x很大,那么将$\sqrt{x+1}-\sqrt{x}$分子有理化,即

$$\sqrt{x+1}-\sqrt{x}=\frac{1}{\sqrt{x+1}+\sqrt{x}},$$

但是要因题而异,如

$$\frac{1}{x}-\frac{1}{x+1}=\frac{1}{x(x+1)},$$

即通分.

例1.4.1 求数$x=7^{15}\times(\sqrt{1+8^{-19}}-1)$的近似值.

解 (1) 直接用MATLAB命令

```
>> x=(7^15)*(sqrt(1+8^(-19))-1)
```

运行后输出结果

```
x = 0
```

问题出现在两个相近的数$\sqrt{1+8^{-19}}$与1相减时,计算机运行程序

```
>>sqrt(1+8^(-19))-1
```

运行后输出结果

```
ans = 0
```

由于计算机硬件只支持有限位机器数的运算,因此在计算中可能引入和传播舍入误差. 因为有效数字的严重损失,导致输出$\sqrt{1+8^{-19}}-1$的结果为0,计算机不能再与数7^{15}继续进行真实的计算,所以,最后输出的结果与x的精确值

不符.

（2）如果化为

$$x=7^{15}\times(\sqrt{1+8^{-19}}-1)=\frac{7^{15}\times 8^{-19}}{\sqrt{1+8^{-19}}+1},$$

再用 MATLAB 命令

```
>> x =(7^15) * ( (8^( -19)) /(sqrt(1 +8^(  19)) +1))
```

运行后输出结果

```
x - 1.6471e -005
```

这是因为 $\sqrt{1+8^{-19}}-1$ 化为 $\frac{8^{-19}}{\sqrt{1+8^{-19}}+1}$ 后，计算机运行程序

```
>> x = (8^( -19)) /(sqrt(1 +8^( -19)) +1)
```

运行后的结果为

```
x =3.4694e -018
```

由于有效数字的损失甚少，所以运算的结果 3.4694×10^{-18} 再与 7^{15} 继续计算，最后输出的结果与 x 的精确值相差无几.

方法 2　对数变换法

如果两个相近的正数 x 与 y 的对数相减时，我们可以变换算法为

$$\log_a x-\log_a y=\log_a\frac{x}{y}\quad(a>0,a\neq1).$$

例 1.4.2　求数 $y=\ln(30-\sqrt{30^2-1})$ 的近似值.

解　（1）直接用 MATLAB 程序

```
>> x =30;x1 = sqrt(x^2 -1)
```

运行后输出结果

```
x1 = 29.9833
```

输入 MATLAB 程序

```
>> x =30; x1 =29.9833;y = log(x -x1)
```

运行后输出结果

```
y = -4.0923
```

（2）因为 $\ln(30-\sqrt{30^2-1})$ 中的 $x=30$ 很大，如果采用倒数变换法

$$z_1=x-\sqrt{x^2-1}=\frac{1}{x+\sqrt{x^2-1}},$$

即

$$\ln(30-\sqrt{30^2-1})=\ln\frac{1}{30+\sqrt{30^2-1}}=-\ln(30+\sqrt{900-1}).$$

输入 MATLAB 程序

```
>> x=30;y=-log(x+sqrt(x^2-1))
```

运行后输出结果

```
y = -4.0941
```

(3) 输入 MATLAB 程序

```
>> x=30; y=log(x-sqrt(x^2-1))
```

运行后输出结果

```
y = -4.0941
```

可见,(2)计算的近似值比(1)的误差小.

方法 3 余弦变换法

如果数 x 接近于零时,我们可以变换算法为

$$1-\cos x=2\sin^2\frac{x}{2},\frac{1-\cos x}{\sin x}=\frac{\sin x}{1+\cos x}.$$

例 1.4.3 利用四位数学用表求 $1-\cos 1°$的近似值.

解 由查表得 $\cos 1°\approx 0.999\ 8$,于是

$$x^*=1-\cos 1°\approx 1-0.999\ 8=0.2\times 10^{-3}=x,$$

且

$$|x^*-x|=|\cos 1°-0.999\ 8|<0.477\ 0\times 10^{-4}<\frac{1}{2}\times 10^{-4},$$

所以近似值 $x=0.000\ 2$ 有一位有效数字.

如果改用公式 $1-\cos 1°=2\sin^2 30'$,查表得 $\sin 30'\approx 0.008\ 7$,于是

$$x^*=1-\cos 1°=2\sin^2 30'\approx 0.151\ 38\times 10^{-3}=x,$$

$$|x^*-x|<0.924\ 8\times 10^{-6}<\frac{1}{2}\times 10^{-5},$$

所以近似值 $x=0.151\ 38\times 10^{-3}$有两位有效数字.

方法 4 泰勒级数法

当$f(x^*)\approx f(x)$时,为了避免有效数位的下降,可以用泰勒级数展开作近似计算

$$f(x^*)-f(x)=f'(x)(x^*-x)+\frac{f''(x)}{2!}(x^*-x)^2+\cdots.$$

如果找不到适合的方法,也可以采用增加有效数位的方法.

1.4.2 避免绝对值太小的数作除数或除数的绝对值远小于被除数的绝对值

从(1.26)式知$f(x,y)=\dfrac{x}{y}$的绝对误差为

$$\left|e^{*}\left(\frac{x}{y}\right)\right| \leqslant \left|\frac{1}{y}\right| |e^{*}(x)| + \left|\frac{x}{y^{2}}\right| |e^{*}(y)|.$$

当$|y|$很小或者$|y| << |x|$时,$f(x,y)$的绝对误差可能很大. 因此不宜将绝对值太小的数作除数,并且应该避免除数的绝对值远小于被除数的绝对值.

1.4.3 避免大数“吃掉”小数的现象

参加计算的数,有时数量级相差很大. 如果不注意采取相应的措施,在它们的加减法运算中,绝对值很小的那个数经常会被绝对值较大的那个数“吃掉”,不能发挥其作用,造成计算结果失真.

例 1.4.4 请在 16 位十进制数值精度计算机上利用软件 MATLAB 计算下面的两个数 $x^{*}=111\ 111\ 111\ 111\ 111+0.1+0.3$ 和 $y^{*}=1\ 111\ 111\ 111\ 111\ 111+0.1+0.3$ 将计算结果与准确值比较,解释计算结果.

解 在 MATLAB 工作窗口输入下面程序

```
>> x =111111111111111 +0.1 +0.3, y =1111111111111111 +0.1 +0.3
```

运行后输出结果

```
x = 1.111111111111114e+014,y =1.111111111111111e+015
```

从输出的结果可以看出,$x=x^{*}$,而 $y \neq y^{*}$. 为什么 y^{*} 仅仅比 x^{*} 多一位 1,而 $y \neq y^{*}$ 呢?

这是因为**计算机进行运算时,首先要把参加运算的数写成绝对值小于 1 而“阶码”相同的数,这一过程称为数的“对阶”**. 在 16 位十进制数值精度计算机上利用软件 MATLAB 计算这两个数,把运算的数 x^{*} 写成浮点规格化形式为

$$x^{*}=0.111\ 111\ 111\ 111\ 111\ 0\times 10^{15}+0.000\ 000\ 000\ 000\ 000\ 1\times 10^{15}+0.000\ 000\ 000\ 000\ 000\ 3\times 10^{15},$$

在 16 位十进制数值精度计算机上,三项的数都表示为小数点后面 16 位数字的数与 10^{15}之积,所以,计算机没有对数进行截断,而是按原来的三个数进行计算. 因此,计算的结果 $x=x^{*}$. 而

$$y^{*}=0.111\ 111\ 111\ 111\ 111\ 1\times 10^{16}+0.000\ 000\ 000\ 000\ 000\ 01\times 10^{16}+0.000\ 000\ 000\ 000\ 000\ 03\times 10^{16}$$

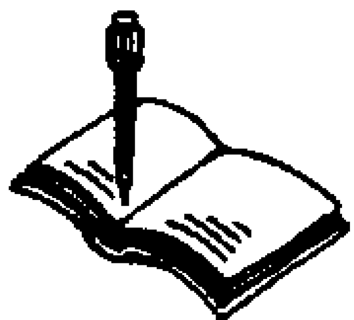
三项的数都表示写成绝对值小于 1 而“阶码”都为 10^{16} 的数以后,第一项的纯小数的小数点后面有 16 位数字. 但是,后两项数的纯小数的小数点后面有 17 位数字,超过了 16 位十进制数值精度计算机的存储量,计算机对后两项的数都进行截断最后一位,即后两项的数都是 16 位机上的零,再进行计算,所以计算结果与实际不符.

一般来说,**多个数相加时,应该将绝对值较小的数相加之后,再依次与绝对值较大的数相加.**

1.4.4 注意简化计算程序,减少运算次数

同样一个计算问题,如果能减少运算次数,不仅可以减少计算量,也能减少误差积累,这是数值计算中必须遵守的原则.

例如,在例1.3.4中为了求方程 $2x^2+x-15=0$ 在(2,3)的一个正根 x^* 的近似值,我们设计三种算法. 虽然,算法2和算法3所产生的迭代序列 $\{x_n\}$ 都收敛到 $x^*=2.5$,但是,算法3只迭代了4次,$\{x_n\}$ 收敛到2.5,且相对误差和绝对误差分别达到了 1.7764×10^{-16} 和 4.4409×10^{-16},然而,算法2迭代了99次,$\{x_n\}$ 才收敛到2.5,且相对误差和绝对误差分别仅仅达到了 3.1877×10^{-9} 和 7.9693×10^{-9}. 造成上述结果的原因是算法2比算法3的初始误差大,并且运算次数多,舍入误差的积累也比较大.



习 题 1.4

1. 求数 $x=4^{15}\times(\sqrt{4+7^{-19}}-2)$ 的近似值.

2. 利用四位数学用表求 $1-\cos 2°$ 的近似值,要求至少有2位有效数字.

3. 请在16位十进制数值精度计算机上利用软件MATLAB计算下面的两个数 $x^*=333\,333\,333\,333\,333+0.1+0.3$ 和 $y^*=33\,333\,333\,333\,333\,333+0.1+0.3$,并将计算结果与准确值比较,解释计算结果.

4. 利用恒等变换,使下列表达式的计算结果比较精确.

(1) $\sqrt{x+\frac{1}{x}}-\sqrt{x-\frac{1}{x}},x>>1$;　(2) $e^{x^2}-1$;　(3) $\frac{1-\cos x}{\sin x}$.

5. 设 $y=\ln x$. 当 $x\approx a\ (a>0)$ 时,如果已知对数 $\ln a$ 的绝对误差限为 $\frac{1}{2}\times10^{-n}$,试估计真数 a 的相对误差限及有效数字位数.

6. 指出下列各题的合理计算途径(对给出具体数据的,请算出结果):

(1) $1-\cos 1°$(三角函数值取四位有效数字);　(2) $\ln(30-\sqrt{30^2-1})$;

(3) $\frac{1-\cos x}{\sin x}$(其中 $|x|$ 充分小);　(4) x^{127}.

7. 设初始数据 x^*,y^* 是相互独立的自变量,x,y 是 x^*,y^* 的近似值,它们的绝对误差分别为 $e^*(x)=x^*-x,e^*(y)=y^*-y$,且绝对值都很小. 证明 x,y 进行加、减、乘、除运算时,初始数据的误差估计和计算结果中产生的误差之间满足关系式(1.18)、(1.19)、(1.22)、(1.23)、(1.26)和(1.27).

1.5 向量和矩阵的范数

在研究方程组近似解的误差估计和迭代法收敛性的过程中,需要对向量和

矩阵的"大小"加以度量,向量范数和矩阵范数正是这种度量指标,在数值分析中起着重要作用.

1.5.1 向量范数与 MATLAB 命令

在3维空间中,将向量 $\boldsymbol{x}=(x_1,x_2,x_3)^{\mathrm{T}}$ 的长度记作 $\|\boldsymbol{x}\|$,通常用 $\|\boldsymbol{x}\|=(\sum_{i=1}^{3}x_i^2)^{\frac{1}{2}}$ 来定义,它具有如下明显的性质:

(1) $\|\boldsymbol{x}\|\geqslant 0$,且仅当 $\boldsymbol{x}=\boldsymbol{0}$(即 $x_1=x_2=x_3=0$)时才有 $\|\boldsymbol{x}\|=0$ 成立;

(2) 对于任何实数 α 有 $\|\alpha\boldsymbol{x}\|=|\alpha|\cdot\|\boldsymbol{x}\|$;

(3) 对任意两个3维向量 $\boldsymbol{x},\boldsymbol{y}$, $\|\boldsymbol{x}+\boldsymbol{y}\|\leqslant\|\boldsymbol{x}\|+\|\boldsymbol{y}\|$(即三角形两边之和不小于第三边).

$\boldsymbol{x}$ 的长度还可以用其他指标来度量,比如定义为 $\|\boldsymbol{x}\|=\max\limits_{1\leqslant x\leqslant 3}|\boldsymbol{x}_i|$,更容易计算. 不难验证,它也具有上面给出的性质.

将3维空间中的这些度量向量长度的指标推广到 n 维(实)向量空间 $\mathbf{R}^n$,并且用指标应满足的性质作为定义,就得到:

定义 1.5 设 $x\in\mathbf{R}^n$,若存在实函数 $N(\boldsymbol{x})=\|\boldsymbol{x}\|$ 满足:

(1) $\|\boldsymbol{x}\|\geqslant 0$,且仅当 $\boldsymbol{x}=\boldsymbol{0}$(指零向量)时 $\|\boldsymbol{x}\|=0$ (正定性);

(2) $\forall\alpha\in\mathbf{R}$, $\|\alpha\boldsymbol{x}\|=|\alpha|\cdot\|\boldsymbol{x}\|$ (齐次性);

(3) $\forall\boldsymbol{y}\in\mathbf{R}^n$, $\|\boldsymbol{x}+\boldsymbol{y}\|\leqslant\|\boldsymbol{x}\|+\|\boldsymbol{y}\|$ (三角不等式),

则称 $\|\boldsymbol{x}\|$ 为**向量 $\boldsymbol{x}$ 的范数**(或模).

定义 1.6 记 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}\in\mathbf{R}^n$,则定义如下:

(1) p **范数** $$\|\boldsymbol{x}\|_p=(\sum_{i=1}^{n}|x_i|^p)^{1/p}\ (1\leqslant p<\infty);\tag{1.28}$$

(2) 2 **范数** $$\|\boldsymbol{x}\|_2=(\sum_{i=1}^{n}x_i^2)^{1/2};\tag{1.29}$$

(3) 1 **范数** $$\|\boldsymbol{x}\|_1=\sum_{i=1}^{n}|x_i|;\tag{1.30}$$

(4) ∞ **范数** $$\|\boldsymbol{x}\|_\infty=\max_{1\leqslant x\leqslant n}|x_i|;\tag{1.31}$$

(5) $-\infty$ **范数** $$\|\boldsymbol{x}\|_{-\infty}=\min_{1\leqslant x\leqslant n}|x_i|,\tag{1.32}$$

并且可以验证它们都满足定义1.5中的条件,所以这种定义是合理的. 另外,我们不难看出,$\|\boldsymbol{x}\|_2$ 和 $\|\boldsymbol{x}\|_1$ 分别是 $\|\boldsymbol{x}\|_p$ 的特例.

一个向量的不同范数虽然通常数值不同,但存在着如下的等价关系:

定理 1.4(向量范数等价性) 设 $\|\boldsymbol{x}\|_\alpha$, $\|\boldsymbol{x}\|_\beta$ 为任意两种范数,则存在常数 $c_1,c_2>0$ 使

$$c_1\|\boldsymbol{x}\|_\alpha\leqslant\|\boldsymbol{x}\|_\beta\leqslant c_2\|\boldsymbol{x}\|_\alpha$$

对任意 $x \in \mathbf{R}^n$ 成立.

由范数的定义和等价性,立即得到向量序列 $\{x^{(k)}\}(k=1,2,\cdots)$ 的收敛性:

定理 1.5 $\lim\limits_{k\to\infty} x^{(k)} = x^*$ 的充要条件是 $\lim\limits_{k\to\infty} \| x^{(k)} - x^* \| = 0$,其中 $\| \cdot \|$ 表示任何一种范数.

向量范数的 MATLAB 命令及功能列入表 1-5:

表 1-5 向量范数的 MATLAB 命令及功能

MATLAB 命令	等价的命令	功 能
Xp = norm (X,P)	sum(abs(X).^P)^(1/P)	输入 X 为向量,输出为 X 的 p 范数 $1\leqslant p<\infty$
X2 = norm(X) 或 X2 = norm(X,2)	sum(abs(X).^2)^(1/2)	输入 X 为向量,输出为 X 的 2 范数
X1 = norm(X,1)	sum(abs(X))	输入 X 为向量,输出为 X 的 1 范数
Xw = norm(X, inf)	max(abs(X))	输入 X 为向量,输出为 X 的 ∞ 范数
Xfw = norm(X, -inf)	min(abs(X))	输入 X 为向量,输出为 X 的 $-\infty$ 范数

例 1.5.1 用 MATLAB 求下列向量的 2 范数、1 范数、∞ 范数、5 范数、$-\infty$ 范数:

(1) $X=(1,-1,2,3)$; (2) $Y=(-0.5,89,0.06,45,-12)^{\mathrm{T}}$.

解 (1) 在 MATLAB 工作窗口输入下列程序

```
>> X = [1, -1,2,3]; X2 = norm(X), X2 = norm(X,2), X1 = norm(X,1),
   Xw = norm(X,inf), Xfw = norm (X, -inf), X5 = norm(X,5)
```

运行后输出如下结果

```
X2 = 3.87298334620742,X1 = 7.00000000000000,
Xw =3,Xfw =1,X5 =3.07961164958130
```

(2) 在 MATLAB 工作窗口输入下列程序

```
>> Y = [ -0.5,89,0.06,45, -12]';Y2 = norm(Y), Y2 = norm(Y,2),
   Y1 = norm(Y,1), Yw = norm(Y,inf),
   Yfw = norm (Y, -inf), Y5 = norm(Y,5)
```

运行后输出如下结果

```
Y2 =1.004502543550787e+002,Y1 =1.465600000000000e+002,
Yw = 89, Yfw = 0.06000000000000,Y5 =89.58135858955995.
```

1.5.2 矩阵范数与 MATLAB 命令

用 $\mathbf{R}^{n\times n}$ 表示 $n\times n$ 矩阵的集合,将向量范数的概念推广到矩阵有下面的

定义.

定义 1.7 设 $A \in \mathbf{R}^{n \times n}$,若存在实函数 $N(A)=\|A\|$ 满足:

(1) $\|A\| \geqslant 0$,当且仅当 $A=\mathbf{0}$ (指零矩阵)时 $\|A\|=0$;

(2) $\forall \alpha \in \mathbf{R}$, $\|\alpha A\|=|\alpha| \cdot \|A\|$;

(3) $\forall B \in \mathbf{R}^{n \times n}$, $\|A+B\| \leqslant \|A\|+\|B\|$;

(4) $\forall B \in \mathbf{R}^{n \times n}$, $\|AB\| \leqslant \|A\| \cdot \|B\|$,

则称 $\|A\|$ 为**矩阵 A 的范数**.

记 $A=(a_{ij})_{n \times n}$,不难验证作为 $\|x\|_2=\left(\sum_{i=1}^{n} x_i^2\right)^{1/2}$ 的简单推广,$\|A\|_F=\left(\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2\right)^{1/2}$ 就是一种符合上述定义的矩阵范数,

定义 1.8 设 $A=(a_{ij})_{n \times n}$,则

$$\|A\|_F=\left(\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2\right)^{1/2}$$

称 $\|A\|_F$ 为矩阵 A 的**弗罗贝尼乌斯(Frobenius)范数**.

由于矩阵和向量常常一起出现在数值分析和计算中,所以希望能引进一种既符合定义 1.7,又与向量范数相联系的矩阵范数.

定义 1.9 如果对于 $x \in \mathbf{R}^n$,$A \in \mathbf{R}^{n \times n}$,有

$$\|Ax\| \leqslant \|A\| \cdot \|x\| \tag{1.33}$$

则称(1.33)式为**矩阵范数与向量范数的相容性条件**.

定义 1.10 对于 $x \in \mathbf{R}^n$,$A \in \mathbf{R}^{n \times n}$和一种向量范数 $\|\cdot\|$,称

$$\|A\|=\max_{x \neq 0} \frac{\|Ax\|}{\|x\|} \tag{1.34}$$

为**矩阵 A 的算子范数**.

可以验证算子范数(1.34)符合定义 1.7,并满足相容性条件(1.33).

显然,算子范数 $\|A\|$ 依于(1.34)式中向量范数的具体含义. 与向量的 2 范数、1 范数、∞ 范数(即(1.29)—(1.31)式)相对应,有

定理 1.6 对于 $x \in \mathbf{R}^n$,$A \in \mathbf{R}^{n \times n}$,矩阵的算子范数为

(1) 2 范数 $$\|A\|_2=\sqrt{\lambda_{\max}(A^{\mathrm{T}}A)}, \tag{1.35}$$

其中 $\lambda_{\max}(A^{\mathrm{T}}A)$ 表示 $A^{\mathrm{T}}A$ 的最大特征值;

(2) 1 范数 $$\|A\|_1=\max_{j} \sum_{i=1}^{n} |a_{ij}|; \tag{1.36}$$

(3) ∞ 范数 $$\|A\|_\infty=\max_{i} \sum_{j=1}^{n} |a_{ij}|. \tag{1.37}$$

(证明略)

定义 1.11 记矩阵 $A \in \mathbf{R}^{n\times n}$ 的特征值为 $\lambda_1,\lambda_2,\cdots,\lambda_n$，称

$$\rho(A) = \max_{1\leq i\leq n} |\lambda_i| \tag{1.38}$$

为 $A \in \mathbf{R}^{n\times n}$ 的谱半径.

谱半径与算子范数的关系由如下定理给出.

定理 1.7 对于 $A \in \mathbf{R}^{n\times n}$，有

$$\rho(A) \leq \|A\|, \tag{1.39}$$

这里 $\|A\|$ 是任何一种矩阵范数.

矩阵范数的 MATLAB 命令列入表 1-6：

表 1-6 矩阵范数的 MATLAB 命令

MATLAB 命令	等价的命令	功 能
X2 = norm(X) 或 X2 = norm(X,2)	max(svd(X))	输入 X 为向量或矩阵，输出为 X 的 2 范数
X1 = norm(X,1)	max(sum(abs(X)))	输入 X 为向量或矩阵，输出为 X 的 1 范数
Xw = norm(X, inf)	max(sum(abs(X')))	输入 X 为向量或矩阵，输出为 X 的∞范数
Xf = norm (X,'fro')	sqrt(sum(diag(X'*X)))	输入 X 为矩阵，输出为 X 的弗罗贝尼乌斯范数
XP = norm(X,P)	(1) 输入 X 为向量，$1\leq p<\infty$，输出为 X 的 p 范数 (2) 输入 X 为矩阵，$p=1,2$，inf 或 'fro'，输出为 X 的 p 范数	
Xpj = max(eig(X))	输入 X 为矩阵，输出为 X 的谱半径	

例 1.5.2 用 MATLAB 求矩阵 A 的 2 范数、1 范数、∞ 范数、弗罗贝尼乌斯范数、谱半径，其中

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & -2 & -3 & -4 \\ 0.1 & 0.2 & 0.3 & 0.5 \\ 5 & 7 & 8 & 9 \end{pmatrix}.$$

解 在 MATLAB 工作窗口输入下列程序

```
>> A = [1,2,3,4;1, -2, -3, -4;0.1,0.2,0.3,0.5;5,7,8,9];
   A2 = norm(A), A2 = norm(A,2), A1 = norm(A,1),
   Aw = norm(A,inf), Af = norm (A,'fro'), Apj = max(eig(A))
```

运行后输出如下结果

```
A2 = 16.46351538590879,A2 =16.46351538590879,
A1 =17.50000000000000,Aw =29,
Af =16.71496335622666,Apj =9.22195075784883.
```

1.5.3 条件数与误差估计及其 MATLAB 命令

定义 1.12 设 $\boldsymbol{A}\in\mathbf{R}^{n\times n}$ 可逆,对于 $\boldsymbol{A}$ 的任意一种算子范数 $\|\boldsymbol{A}\|$,称

$$\operatorname{cond}(\boldsymbol{A})=\|\boldsymbol{A}\|\cdot\|\boldsymbol{A}^{-1}\| \tag{1.40}$$

为矩阵 $\boldsymbol{A}$ 的条件数. 对于 $\|\boldsymbol{A}\|_2,\|\boldsymbol{A}\|_1,\|\boldsymbol{A}\|_\infty$,相应的有 $\operatorname{cond}_2(\boldsymbol{A})$,$\operatorname{cond}_1(\boldsymbol{A})$,$\operatorname{cond}_\infty(\boldsymbol{A})$

定理 1.8 条件数有如下性质:

(1) $\operatorname{cond}(\boldsymbol{A})\geqslant 1$;

(2) 对于 $\alpha(\neq 0)\in\mathbf{R},\operatorname{cond}(\alpha\boldsymbol{A})=\operatorname{cond}(\boldsymbol{A})$;

(3) 对于正交矩阵 $\boldsymbol{Q}\in\mathbf{R},\operatorname{cond}(\boldsymbol{QA})=\operatorname{cond}(\boldsymbol{AQ})=\operatorname{cond}(\boldsymbol{A})$.

对于方程组 $\boldsymbol{Ax}=\boldsymbol{b}$,当 $\boldsymbol{b}$ 有误差 $\delta\boldsymbol{b}$ ($\boldsymbol{A}$ 不变)或 $\boldsymbol{A}$ 有误差 $\delta\boldsymbol{A}$($\boldsymbol{b}$ 不变)时,解 $\boldsymbol{x}$ 相应的有误差 $\delta\boldsymbol{x}$,其大小可由下式估计

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\leqslant\operatorname{cond}(\boldsymbol{A})\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}, \tag{1.41}$$

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}+\delta\boldsymbol{x}\|}\leqslant\operatorname{cond}(\boldsymbol{A})\frac{\|\delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|}. \tag{1.42}$$

所以矩阵的条件数可视为矩阵病态程度的一种度量,条件数越大,病态越严重,引起方程组解的误差可能越大.

条件数的 MATLAB 命令列入表 1-7:

表 1-7 条件数的 MATLAB 命令

MATLAB 命令	功 能
Xc2 = cond (X) 或 Xc2 = cond (X,2)	输入 $\boldsymbol{X}$ 为矩阵,输出为 $\boldsymbol{X}$ 的 2 范数的条件数
Xc1 = cond (X,1)	输入 $\boldsymbol{X}$ 为矩阵,输出为 $\boldsymbol{X}$ 的 1 范数的条件数
Xcw = cond (X, inf)	输入 $\boldsymbol{X}$ 为矩阵,输出为 $\boldsymbol{X}$ 的∞ 范数的条件数
Xcf = cond (X,'fro')	输入 $\boldsymbol{X}$ 为矩阵,输出为 $\boldsymbol{X}$ 的弗罗贝尼乌斯范数条件数
Xcp = cond(X,P)	输入 $\boldsymbol{X}$ 为矩阵,输出为 $\boldsymbol{X}$ 的 p 条件数,其中 p =1,2,inf,或'fro.'
Xcr = rcond(X)	输入 $\boldsymbol{X}$ 为矩阵,输出为 $\boldsymbol{X}$ 的 1 范数的条件数倒数的估计值. 如果 $\boldsymbol{X}$ 条件很好,rcond($\boldsymbol{X}$)在 1.0 附近;如果 $\boldsymbol{X}$ 条件不好,rcond($\boldsymbol{X}$)在 EPS 附近

更多的信息请看 COND, CONDEST, NORMEST.

help ss/norm.m, help lti/norm.m, help frd/norm.

例 1.5.3 用 MATLAB 求矩阵 $\boldsymbol{A}$ 的 2 范数条件数、1 范数条件数、∞ 范数条件数、弗罗贝尼乌斯范数条件数、条件数倒数的估计值,其中

$$\boldsymbol{A}=\begin{pmatrix}11 & 2 & 3 & 4\\ 7 & -2 & -3 & -4\\ 0.1 & 0.2 & 0.3 & 0.5\\ 5 & 7 & 8 & 9\end{pmatrix}.$$

解 在 MATLAB 工作窗口输入下列程序

```
>> A=[11,2,3,4;7,-2,-3,-4;0.1,0.2,0.3,0.5;5,7,8,9];
   Ac2 = cond (A), Ac2 = cond (A,2), Ac1 = cond (A,1),
   Acw= cond (A,inf), Acf = cond (A,'fro'), Acr=rcond (A)
```

运行后输出如下结果

```
Ac2 =4.404563972867626e+002,Ac1 =9.239999999999991e+002,
Acw =6.901999999999993e+002,Acf =5.225127575476023e+002,
Acr =0.00108225108225
```

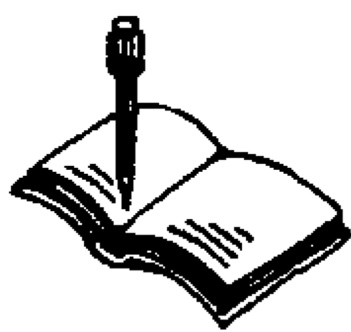
例 1.5.4 设 $\boldsymbol{A}=\begin{pmatrix}10 & 7 & 8 & 7\\ 7 & 5 & 6 & 5\\ 8 & 6 & 10 & 9\\ 7 & 5 & 9 & 10\end{pmatrix}$,当 $\boldsymbol{b}=(32,23,33,31)^{\mathrm{T}}$ 有微小误差 $\delta\boldsymbol{b}=(0.1,-0.1,0.1,-0.1)^{\mathrm{T}}$ 时,估计方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 解的变化.

解 取 2 范数和 2 条件数,用 MATLAB 得到 $\dfrac{\|\delta\boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2}\approx 0.0033$, $\mathrm{cond}_2(\boldsymbol{A})\approx 3000$,由(1.41)式估计 $\dfrac{\|\delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}\leqslant 10$,即(相对)误差放大了约 3 000 倍.

实际上用 MATLAB 也容易得到

$$\boldsymbol{x}=(1,1,1,1)^{\mathrm{T}},\boldsymbol{x}+\delta\boldsymbol{x}=(9.2,-12.6,4.5,-2.1)^{\mathrm{T}},\frac{\|\delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}\approx 8,$$

可见估计相差不远.



习 题 1.5

1. 设 $\boldsymbol{A}=\begin{pmatrix}10 & 7 & 8 & 7\\ 7 & 5 & 6 & 5\\ 8 & 6 & 10 & 9\\ 7 & 5 & 9 & 10\end{pmatrix}$,当 $\boldsymbol{b}=(32,23,33,31)^{\mathrm{T}}$ 有微小误差 $\delta\boldsymbol{b}=(0.1,-0.1,0.1,$

$-0.1)^T$ 时,估计方程组 $Ax=b$ 解的变化.

2. 用 MATLAB 求矩阵 A 的 2 范数条件数、1 范数条件数、∞ 范数条件数、弗罗贝尼乌斯范数条件数、条件数倒数的估计值,其中

$$A=\begin{pmatrix}10 & 2 & 3 & 4\\ 15 & -2 & -3 & -4\\ 6 & 0.2 & 0.3 & 0.5\\ 5 & 7 & 8 & 9\end{pmatrix}.$$

3. 用 MATLAB 求第 2 题中矩阵 A 的 2 范数、1 范数、∞ 范数、弗罗贝尼乌斯范数、谱半径.

4. 用 MATLAB 求下列向量的 2 范数、1 范数、∞ 范数、5 范数、−∞ 范数:

(1) $X=(1,-1,2,5)$; (2) $Y=(-0.4,89,0.06,45,-12)^T$.

5. 线性方程组求解与性态讨论.

求 $Ax=b$ 的解向量 x,其中

$$A=\begin{pmatrix}10 & 7 & 8 & 7\\ 7 & 5 & 6 & 5\\ 8 & 6 & 10 & 9\\ 7 & 5 & 10 & 9\end{pmatrix},\quad b=\begin{pmatrix}32\\ 23\\ 33\\ 31\end{pmatrix}.$$

然后把 b 扰动为 $\hat{b}=(32.01,22.99,33.01,30.99)^T$,再求解 $A\hat{x}=\hat{b}$. 计算 $\|\hat{x}-x\|/\|x\|$(使用 1 范数或∞ 范数),讨论方程组性态.

第二章 非线性方程(组)的数值解法

本章主要介绍方程根的有关概念,求方程根的步骤,确定根的初始近似值的方法(作图法,逐步搜索法等),求根的方法(二分法,迭代法,牛顿法,割线法,米勒(Müller)法和迭代法的加速等)及其 MATLAB 程序,求解非线性方程组的方法及其 MATLAB 程序.

2.1 求方程(组)的根及其 MATLAB 命令

2.1.1 方程的概念

在科学和工程技术计算中的许多问题常归结为求解方程

$$f(x)=0, \tag{2.1}$$

当函数 $f(x)$ 是一次多项式时,称方程(2.1)为**线性方程**;若方程(2.1)中包含三角函数、指数函数等超越函数(如 $\sin x, e^x, \ln x$ 等)时,则称方程(2.1)为**超越方程**. 它与 $n\ (\geqslant 2)$ 次代数方程一起统称**非线性方程**. 如果有 x^* 使得 $f(x^*)=0$,则称 x^* 为方程(2.1)的**根**,或称为 $f(x)$ 的**零点**. 设有正整数 m 使得

$$f(x)=(x-x^*)^m g(x)$$

且 $g(x^*)\neq 0$,则当 $m\geqslant 2$ 时,称 x^* 为方程(2.1)的 m 重根;当 $m=1$ 时,称 x^* 为方程(2.1)的单根.

关于 n 次代数方程

$$a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n=0, \tag{2.2}$$

当 $n\leqslant 4$ 时,有求根公式. 当 $n>4$ 时,方程就没有现成的求根公式了. 但是由数学基本定理知,n 次代数方程一定有 n 个根,包括复根,当然重根应按重数计算根的个数. 对于超越方程不仅没有一般的求根公式,而且若只依据方程本身,那么连有没有根、有几个根也难以判断. 然而,在实际应用问题中,不一定要求得到根的解析式,只要求获得具有一定精确程度的近似值(即数值解)就可以了. 求解非线性方程的数值解一般可分如下三步进行:

1. 判断根的存在性. 即方程有没有根? 如果有根,有几个根?
2. 确定根的近似位置. 即求出根所在的区间或确定根的初始近似值.
3. 根的精确化. 即按某种方法将初始近似值逐步精确化,直到满足所要求的精确度.

2.1.2 求解方程(组)的 solve 命令

求方程 $f(x)=q(x)$ 的根可以用 MATLAB 命令

```
>> x = solve('方程 f(x) = q(x)','待求符号变量 x')
```

求方程组 $f_i(x_1,\cdots,x_n)=q_i(x_1,\cdots,x_n)\ (i=1,2,\cdots,n)$ 的根可以用 MATLAB 命令

```
>> E1 = sym('方程 f1(x1,...,xn) = q1(x1,...,xn)');
............
En = sym('方程 fn(x1,...,xn) = qn(x1,...,xn)');
[x1,x2,...,xn] = solve(E1,E2,...,En, x1,...,xn)
```

例 2.1.1 解方程 $8x^9+17x^3-3x=-1$.

解 在 MATLAB 命令窗口输入命令

```
>> x = solve('8 * x^9 +17 * x^3 -3 * x = -1')
```

则运行后输出此方程的根为

```
x =
[ -.95778266908844934303923445046258 - .59070709534579563538068568150441 * i]
[ -.95778266908844934303923445046258 + .59070709534579563538068568150441 * i]
[                                      -.53275697173102083274354955617700]
[ -.62165529622636773563391338820961e -2 -1.1576656136411973167991239807601 * i]
[ -.62165529622636773563391338820961e -2 +1.1576656136411973167991239807601 * i]
[    .26762926822201693804563287259316 - .19580612317589362415618068133297 * i]
[    .26762926822201693804563287259316 + .19580612317589362415618068133297 * i]
[    .96274843969420649872171548984002 - .57475793354361098651731421962321 * i]
[    .96274843969420649872171548984002 + .57475793354361098651731421962321 * i]
```

例 2.1.2 解下列方程,并且求根的近似值,要求精度为 10^{-14}.

(1) $\sin(\cos(2x^3))=0$;　　(2) $a\sin(\cos 4)+5\tan(a+3)=0$;

(3) $a\sin(\cos x)+b\tan a=0$.

解 只需在 MATLAB 工作窗口输入命令

```
>> x1 = solve('sin(cos(2 * x^3)) = 0','x')
  a1 = solve('a * sin(cos(4)) +5 * tan(a +3) = 0','a')
  x2 = solve('a * sin(cos(x)) +b * tan(a) = 0','x')
```

则运行后输出三个方程的根如下

```
x1 =
[                                        1/2 * 2^(1/3) * pi^(1/3)]
[ -1/4 * 2^(1/3) * pi^(1/3) +1/4 * i * 3^(1/2) * 2^(1/3) * pi^(1/3)]
[ -1/4 * 2^(1/3) * pi^(1/3) -1/4 * i * 3^(1/2) * 2^(1/3) * pi^(1/3)]
a1 =
```

```
-3.3911693397183601737264718371364
x2 =
pi - acos(asin(tan(a) * b/a))
```

若想求精度为 10^{-14} 的根的近似值,首先需要选中 MATLAB 工作窗口中的下拉菜单 File 的子菜单 Preferences,单击鼠标左键,在 General 对话框的 Double Formal 中选 long;然后,在 MATLAB 工作窗口依次输入程序

```
>> x1 = 1/2 * 2^(1/3) * pi^(1/3),
 x2 = -1/4 * 2^(1/3) * pi^(1/3) +1/4 * i * 3^(1/2) * 2^(1/3) * pi^(1/3),
 x3 = -1/4 * 2^(1/3) * pi^(1/3) -1/4 * i * 3^(1/2) * 2^(1/3) * pi^(1/3)
```

运行后即可得到近似值

```
x1 =0.92263507432201
x2 = -0.46131753716101 + 0.79902541278541i
x3 = -0.46131753716101 - 0.79902541278541i
```

MATLAB 系统本来只能做数值计算,并没有符号运算的功能,符号运算工具箱(Symbolic Math Toolbox)则扩充了 MATLAB 这方面的功能,它是由 Maple 软件的核心来完成的.用 solve 命令求方程的根的方法有一个缺点,它不能求出周期函数 $f(x)$ 对应的方程 $f(x)=0$ 的全部根(如 $\sin x=0$,用命令 >> x = solve ('sin(x) =0 '),运行后输出一个根 $x = 0$).

例 2.1.3 解超越方程组 $\begin{cases} x^x - 4 = 0, \\ 2xy + x = 1. \end{cases}$

解 在 MATLAB 命令窗口输入命令

```
>> E1 = sym('x^x - 4 =0 '); E2 = sym('2 * x * y + x =1 ');
 [x,y] = solve(E1,E2)
 x1 = double (x),y1 = double (y)
```

则运行后输出超越方程组精确解 x,y 和近似解 x_1,y_1 如下

```
x = log(4)/lambertw(log(4))
y = -1/2 * (log(4) - lambertw(log(4)))/log(4)
x1 =2
y1 = -0.2500
```

例 2.1.4 解非线性方程组 $\begin{cases} x + y + z = u, \\ 2xz - yu = -1, \\ (x+y)^2 - z = u, \\ xu + yz = 4. \end{cases}$

解 在 MATLAB 命令窗口输入命令

```
>> E1 = sym('x + y + z = u'); E2 = sym('2 * x * z - y * u = -1 ');
  E3 = sym('(x + y)^2 - z = u');
```

```
E4 = sym('x*u+y*z =4'); [x,y,z,u] = solve(E1,E2,E3,E4),
x1 = double (x),y1 = double (y), z1 = double (z), u1 = double (u)
```

则运行后输出方程组精确解 x,y,z,u(略)和近似解 x_1,y_1,z_1,u_1 如下

```
x1 =
3.00000000000000
0.09943200266874
-1.77193822355659 + 0.46109615509798i
-1.77193822355659 -0.46109615509798i
y1 =
1.00000000000000
23.60367144267526
0.25371983421793 -0.42474610911316i
0.25371983421793 +0.42474610911316i
z1 =
1.00000000000000
-23.43369830490221
-1.00537306977112 - 1.40752182928383i
-1.00537306977112 + 1.40752182928383i
u1 =
1.00000000000000
-0.07054113510432
 1.02028498800340 +2.29336409349498i
-1.02028498800340 -2.29336409349498i
```

例 2.1.5 解非线性方程组 $\begin{cases}\dfrac{x^2}{z}+y^2=0,\\ y+x=1,\\ z^2+5z=9.\end{cases}$

解 在 MATLAB 命令窗口输入命令

```
>> [x,y,z] = solve('(x^2)/z +y^2 =0','x +y =1','z^2 +5*z -9')
x1 = double (x), y1 = double (y), z1 = double (z)
```

则运行后输出非线性方程组精确解 x,y,z 和近似解 x_1,y_1,z_1(略).

例 2.1.6 解非线性方程组 $\begin{cases}x^2+4xy+z=0,\\ 3yz+x=3,\\ z^2+5\sin z-y=0.\end{cases}$

解 在 MATLAB 命令窗口输入命令

```
>> [x,y,z] = solve('x^2 +4*x*y+z=0','x+3*y*z =3','z^2 +5
```

```
*sin(z)-y=0')
```

则运行后输出非线性方程组精确解 x,y,z 和近似解 x_1,y_1,z_1(略).

2.1.3 求解多项式方程(组)的 roots 命令

如果 $f(x)$ 为多项式,则可分别用如下命令求方程 $f(x)=0$ 的根,或求导数 $f'(x)$(见表 2-1).

表 2-1 求解多项式方程(组)的 roots 命令

命令	功能
xk =roots(fa)	输入多项式 $f(x)$ 的系数 fa(按降幂排列),运行后输出 xk 为 $f(x)=0$ 的全部根.
dfa=polyder(fa)	输入多项式 $f(x)$ 的系数 fa(按降幂排列),运行后输出 dfa 为多项式 $f(x)$ 的导数 $f'(x)$ 的系数.
dfx=poly2sym(dfa)	输入多项式 $f(x)$ 的导数 $f'(x)$ 的系数 dfa(按降幂排列),运行后输出 dfx 为多项式 $f(x)$ 的导数 $f'(x)$.

例 2.1.7 解方程 $8x^9+17x^3-3x=-1$,并且求 $f(x)=8x^9+17x^3-3x+1$ 的导数.

解 (1) 首先将已知方程化为 $8x^9+17x^3-3x+1=0$,在 MATLAB 工作窗口输入命令

```
>> fa=[8,0,0,0,0,0,17,0,-3,1]; xk=roots(fa)
```

运行后可得全部根(略).

(2) 如果再求 $f(x)=8x^9+17x^3-3x+1$ 的导数,只要在 MATLAB 工作窗口输入程序

```
>> fa=[8,0,0,0,0,0,17,0,-3,1]; dfa=polyder(fa)
```

运行后屏幕显示多项式 $f(x)$ 的导数的系数如下

```
dfa =   72   0   0   0   0   0   51   0   -3
```

如果再输入命令

```
>> dfx=poly2sym(dfa)
```

运行后屏幕显示多项式 $f(x)$ 的导数如下

```
dfx =72*x^8+51*x^2-3
```

即 $f'(x)=72x^8+51x^2-3$.

这两类求方程的根的方法都有缺点,其中 MATLAB 命令 roots(fa)只能求 $f(x)$ 为多项式时方程 $f(x)=0$ 的根,而命令 solve('方程') 不能求出周期函数 $f(x)$ 对应的方程 $f(x)=0$ 的全部根,所以,我们下面介绍求方程根的近似值的方法.

2.1.4 求解方程(组)的 fsolve 命令

如果非线性方程(组)是多项式形式,求这样方程(组)的数值解可以直接调用上面已经介绍过的 roots 命令.如果非线性方程(组)是含有超越函数,则无法使用 roots 命令,需要调用 MATLAB 系统中提供的另一个程序 fsolve 来求解.当然,程序 fsolve 也可以用于多项式方程(组),但是它的计算量明显比 roots 命令的大.

fsolve 命令使用最小二乘法(least squares method)解非线性方程(组)

$$\boldsymbol{F}(\boldsymbol{X})=\boldsymbol{0}$$

的数值解,其中 $\boldsymbol{X}$ 和 $\boldsymbol{F}(\boldsymbol{X})$ 可以是向量或矩阵.此种方法需要尝试着输入解 $\boldsymbol{X}$ 的初始值(向量或矩阵)$\boldsymbol{X}_0$,即使程序中的迭代序列收敛,也不一定收敛到 $\boldsymbol{F}(\boldsymbol{X})=\boldsymbol{0}$ 的根(见例 2.1.8).

fsolve 的调用格式: X = fsolve(F,X0)

输入函数 $F(x)$ 的 M 文件名和解 $\boldsymbol{X}$ 的初始值(向量或矩阵)$\boldsymbol{X}_0$,尝试着解方程(组)$\boldsymbol{F}(\boldsymbol{X})=\boldsymbol{0}$,运行后输出 $\boldsymbol{F}(\boldsymbol{X})=\boldsymbol{0}$ 解的估计值(向量或矩阵)$\boldsymbol{X}$.

要了解更多的调用格式和功能请输入:help fsolve,查看说明.

例 2.1.8 求方程 $8x^9+17x^3-3x=-1$ 的一个实根.

解 (1) 首先将已知方程化为 $8x^9+17x^3-3x+1=0$,然后在 MATLAB 编辑窗口建立函数子程序,并取名为 Fun1.m 保存,即

```
function F = Fun1(x)
         F = 8 * x^9 + 17 * x^3 - 3 * x + 1;
```

(2) 如果取初始值 $X_0=-0.5$,在 MATLAB 工作窗口输入命令

```
>> X0 = -0.5; X = fsolve('Fun1',X0), F = Fun1(X)
```

运行后可得方程 $8x^9+17x^3-3x=-1$ 的一个实根为

```
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.
X = -0.53275697320556 %初始点附近的一个实根
F = -1.760984380538844e-008 % F是函数F在实根X处的函数值,以此反
映根X是否满足方程的情况.
```

将这个实根 $X=-0.532\ 756\ 973\ 205\ 56$ 与例 2.1.1 的结果 $x_k=-0.532\ 756\ 971\ 731\ 02$ 比较,精确到小数点后 8 位.

(3) 如果取初始值 $X_0=2.5$,在 MATLAB 工作窗口输入命令

```
>> X0 = 2.5; X = fsolve('Fun1',X0), F = Fun1(X)
```

运行后屏幕显示运行结果如下

```
Optimizer appears to be converging to a minimum that is not a root:
Sum of squares of the function values is > sqrt(options.TolFun).
```

```
Try again with a new starting point.
X = 0.24250078639990
F = 0.51495196602749
```

程序提示最优化程序显现收敛与最小值,但不是此方程的根,建议再输入新的初始值 X_0.

例 2.1.9 求方程组 $\begin{cases} x^3 - y^2 = 0, \\ e^{-x} - y = 0 \end{cases}$ 的一个实根.

解 (1) 首先在 MATLAB 编辑窗口建立函数子程序,然后取名为 Fun2.m 并保存,即

```
function F = Fun2(X)
x = X(1);y = X(2); F(1) = x^3 - y^2; F(2) = exp( - x) - y;
```

(2) 如果取初始值向量 $X_0 = (1,1)$,在 MATLAB 工作窗口输入命令

```
>> X0 = [1,1]; X = fsolve('Fun2',X0), F = Fun2(X)
```

运行后屏幕显示运行结果如下

```
Optimization terminated successfully:
First - order optimality is less than options.TolFun.
X = 0.64884423610894   0.52264945514596
F = 1.0e - 006 *
                    0.21955069023916   0.03207949006434
```

程序提示:最优化成功地结束.

例 2.1.10 求方程组 $\begin{cases} x + y + z = 6, \\ x + yz + zx = 8, \\ e^{-x} + \lg y + z = 2 \end{cases}$ 的一个实根.

解 (1) 首先在 MATLAB 编辑窗口建立函数子程序,然后取名为 Fun3.m 并保存,即

```
function F = Fun3(X)
x = X(1);y = X(2);z = X(3); F(1) = x + y + z - 6;
F(2) = x + y * z + z * x - 8; F(3) = exp( - x) + log(y) + z - 2;
```

(2) 如果取初始值向量 $X_0 = (1,1,1)$,在 MATLAB 工作窗口输入命令

```
>> X0 = [1,1,1]; X = fsolve('Fun3',X0), F = Fun3(X)
```

运行后屏幕显示如下

```
Optimization terminated successfully:
First - order optimality is less than options.TolFun.
X = 2.61316848511027 2.28766514388267 1.09916637100706
F = 1.0e - 012 *
    0.00355271367880  - 0.18829382497643  - 0.08659739592076
```

例 2.1.11 求方程 $\sin 3x = 0$ 的两、三个实根.

解 (1) 如果取初始值 $X_0=(1,4)$，在 MATLAB 工作窗口输入命令

```
>> fun = inline('sin(3 * X)');
   X = fsolve(fun,[1 4],optimset('Display','off')), F = fun(X)
```

运行后屏幕显示运行结果如下

```
X =  1.04719755119660  4.18879020478655
F =  1.0e-012 *
        -0.00032162452994  0.46136291952441
```

(2) 如果取初始值 $X_0=(2,3,4)$，在 MATLAB 工作窗口输入命令

```
>> Fun = inline('sin(3 * X)');
   X = fsolve( Fun,[2 3 4],optimset('Display','iter')),
F = Fun(X)
```

运行后屏幕显示结果如下

Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	4	0.535825		1.36	1
1	8	0.00564029	0.277607	0.208	1
2	12	1.26072e-008	0.025087	0.000336	1
3	16	2.13971e-025	3.74273e-005	1.38e-012	1

```
Optimization terminated successfully:
First-order optimality is less than options.TolFun.
X =  2.09439510239320  3.14159265358980  4.18879020478655
F =  1.0e-012 *
     0.00064324905987  -0.03338338590886  0.46136291952441
```



习 题 2.1

1. 用两种方法解方程 $\sin 5x+\cos 2x=-0.31$. 如何一次求出此方程的四个根.
2. 用三种方法解方程 $9x^{11}-12x^8+x^5-3x^2-12=0$.
3. 用 MATLAB 方法求函数 $f(x)=9x^{11}-12x^8+x^5-3x^2-12$ 的导数 $f'(x)$.
4. 求方程组 $\begin{cases}x+y+z=6,\\ x-y+z=2,\\ x^2+y^2-z=2\end{cases}$ 的一个实根，初始点取(2,2,2).

5. 求方程组$\begin{cases}7\sin x+2\cos y=10x,\\7\cos x-2\sin y=10y\end{cases}$的一个实根,初始点取(0.5,0.5).

6. 求方程 $x-\sin x-\ln x=1$ 的两个实根,初始点取(0,2).

2.2 搜索根的方法及其 MATLAB 程序

求解非线性方程根的近似值时,首先需要判断方程有没有根? 如果有根,有几个根? 如果有根,需要搜索根所在的区间或确定根的初始近似值(简称初始值). 搜索根的近似位置的常用方法有三种:作图法、逐步搜索法和二分法等,使用这些方法的前提是高等数学中的零点定理.

定理 2.1(零点定理) 设函数 $f(x)$ 在闭区间 $[a,b]$ 上连续,且 $f(a)\cdot f(b)<0$,那么在开区间 (a,b) 内至少有一点 ξ,使 $f(\xi)=0$.

2.2.1 作图法及其 MATLAB 程序

作图法就是作出函数 $y=f(x)$ 的粗略图形,观察它与 x 轴的交点的位置. 如果根的计算过程属于一个非常庞大的工程,那么可以采用作图法. 通过对图 $y=f(x)$ 进行观察,并根据它的性质(凹凸性、单调性、斜率、振荡性、极值和拐点等)可以做出重要判断. 更重要的是,如果图中的对应点存在,它们可以被分析,并且可以用来决定根的近似值的位置. 这些近似值可以作为求根算法的初始值.

作函数的图形的方法很多,如用计算机软件的图形功能画图,或用高等数学中导数应用作图,或用初等数学的函数叠加法作图等. 下面介绍两种作图程序.

作函数 $y=f(x)$ 在区间 $[a,b]$ 的图形的 MATLAB 程序一

```
x = a:h:b; % h 是步长
y = f(x); plot(x,y)
grid, gtext('y = f(x)')
```

说明:(1) 此程序在 MATLAB 的工作区输入,运行后即可出现函数 $y=f(x)$ 的图形. 此图形与 x 轴交点的横坐标即为所要求的根的近似值.

(2) 区间 $[a,b]$ 的两个端点的距离 $b-a$ 和步长 h 的绝对值越小,图形越精确.

作函数 $y=f(x)$ 在区间 $[a,b]$ 上的图形的 MATLAB 程序二

将 $y=f(x)$ 化为 $h(x)=g(x)$,其中 $h(x)$ 和 $g(x)$ 是两个相等的简单函数

```
x = a:h:b; y1 = h(x); y2 = g(x);
plot(x, y1, x, y2)
grid,gtext('y1 = h(x),y2 = g(x)')
```

说明:此程序在 MATLAB 的工作区输入,运行后即可出现函数 $y_1=h(x)$ 和 $y_2=g(x)$ 的图形. 两图形交点的横坐标即为所要求的根的近似值.

下面举例说明如何用计算机软件 MATLAB 的图形功能作图.

例 2.2.1 利用作图法判断方程 $2x\sin x-3=0$ 在闭区间[-10,10] 上有几个实根,并确定实根的近似位置.

解 方法 1 因为函数 $f(x)=2x\sin x-3$ 是偶函数,所以只需画在[0,10]上的图形即可.

在 MATLAB 工作窗口输入下列程序

```
>>x=0:0.001:10; f=2.*x.*sin(x)-3;plot(x,f)
grid, gtext('f=2xsin(x)-3')
```

运行后可画出函数 $f(x)=2x\sin x-3$ 的图像,见图 2-1(a). 因为在闭区间[0,10]上曲线 $f(x)$ 与 x 轴有四个交点,即方程有 4 个根. 所以,在闭区间[-10,10]上有 8 个实根,它们的近似位置依次为 ±1.5, ±2.5, ±6.5, ±9.4.

方法 2 将方程 $2x\sin x-3=0$ 化为 $2\sin x=3/x$. 因为 $y_1=2\sin x, y_2=3/x$ 都是奇函数,所以只需画在[0,10]上的图形即可.

在 MATLAB 工作窗口输入下列程序

```
>>x=0.8:0.001:10; y1=2.*sin(x);y2=3./x ;
  plot(x, y1, x, y2), grid
  gtext('y1=2sin(x)和 y2=3/x 的图形')
```

运行后可画出函数 $y_1=2\sin x, y_2=3/x$ 的图像,见图 2-1(b). 因为在闭区间[0,10]上两图形有四个交点,它们的横坐标即为所要求的根的近似值. 所以,在闭区间[-10,10] 上有 8 个实根,它们的近似位置依次为 ±1.5, ±2.5, ±6.5, ±9.4.

例 2.2.2 利用作图法判断方程 $4x^5-8x^4-26x^3+30=0$ 是否有实根,并确定实根的近似位置.

解 (1)首先用 MATLAB 程序作函数 $y=4x^5-8x^4-26x^3+30$ 的图形.

根据函数的性质,取 x 的范围从 -50 000 到 30 000 (即取 x 较大的范围,以免漏掉根),步长 $h=10$,用下面的程序

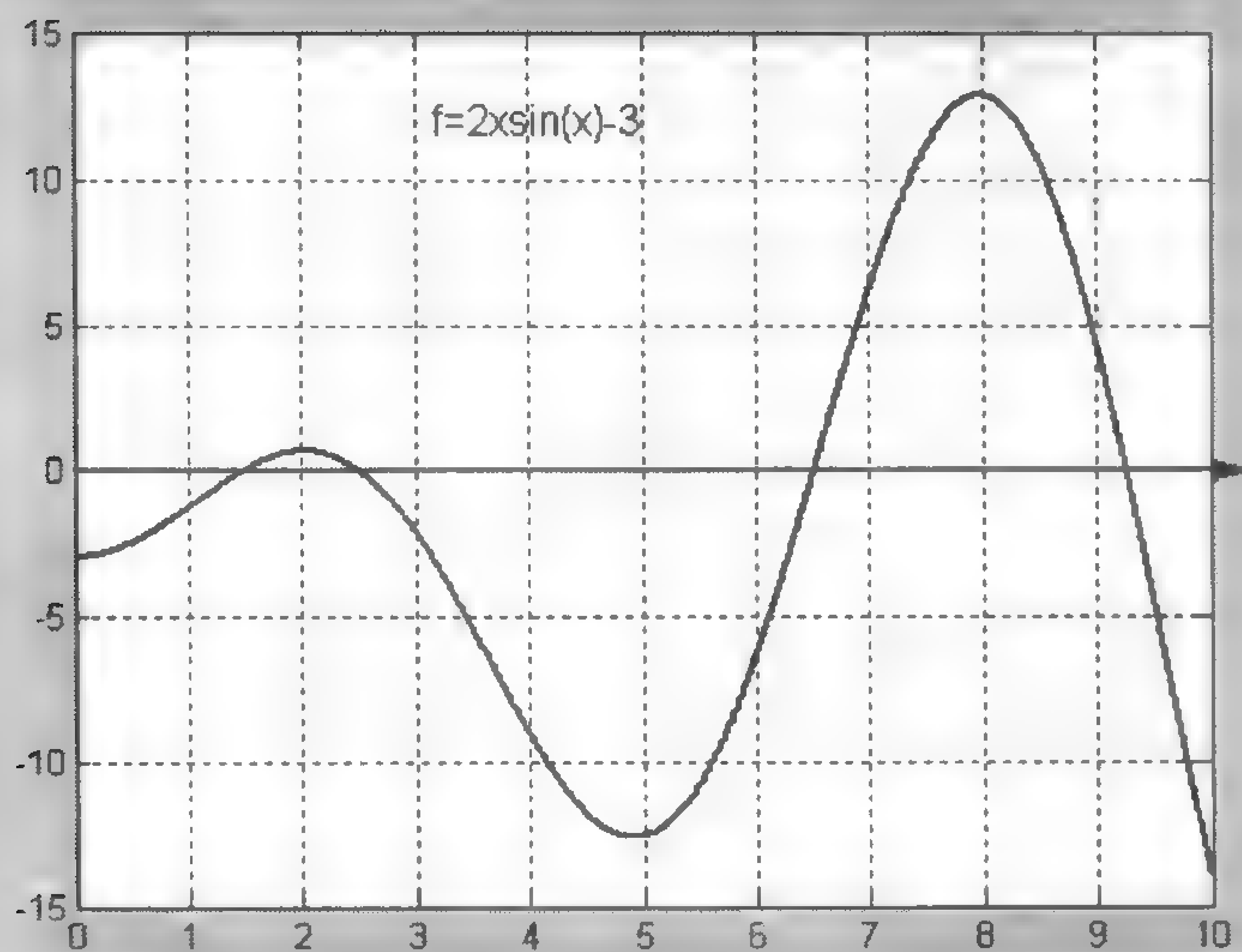
```
>>x=-50000:10:30000;y=4.*x.^5-8.*x.^4-26.*x.^3+30;
  plot(x,y), grid,gtext('y=4x^5-8x^4-26x^3+30')
```

运行后可画出函数 $y=4x^5-8x^4-26x^3+30$ 的图像,即图 2-2(a).

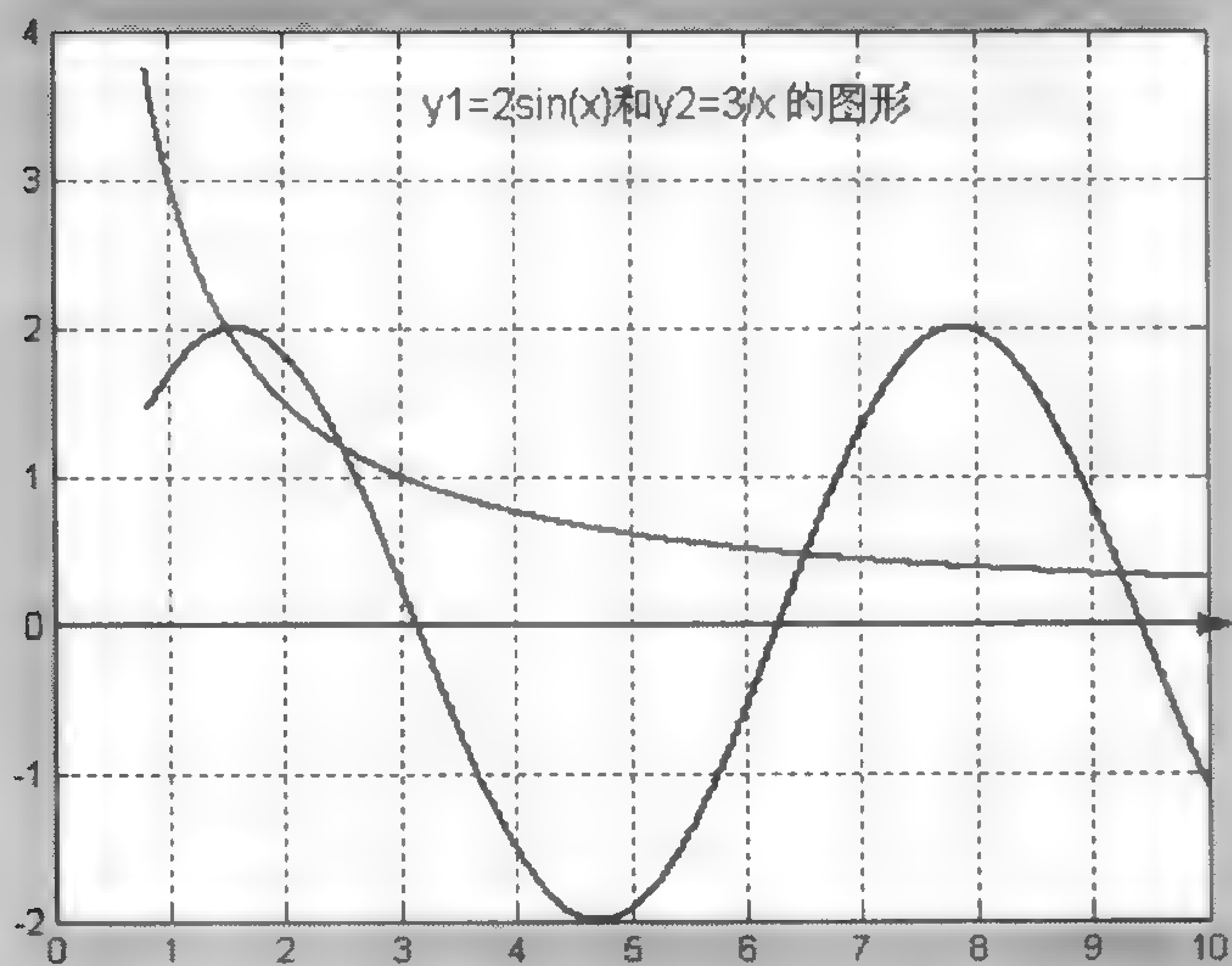
为了清楚地看出该曲线与 x 轴的交点,经过适当调整 x 的范围,取 x 从 -4 到 4,步长 $h=0.1$, 得到图 2-2(b).

(2) 再确定根的个数和范围.

从图 2-2(b)中的曲线与 x 轴的交点可以看出,在图形所表示的范围内,该方程有 3 个实根,一个根为 1,另外 2 个根大约分别位于 $x=-1.91, 3.71$ 附近. 如果我们要得到含根的更精确的区间,只要分别在上面 3 个近似根附近取 x 的

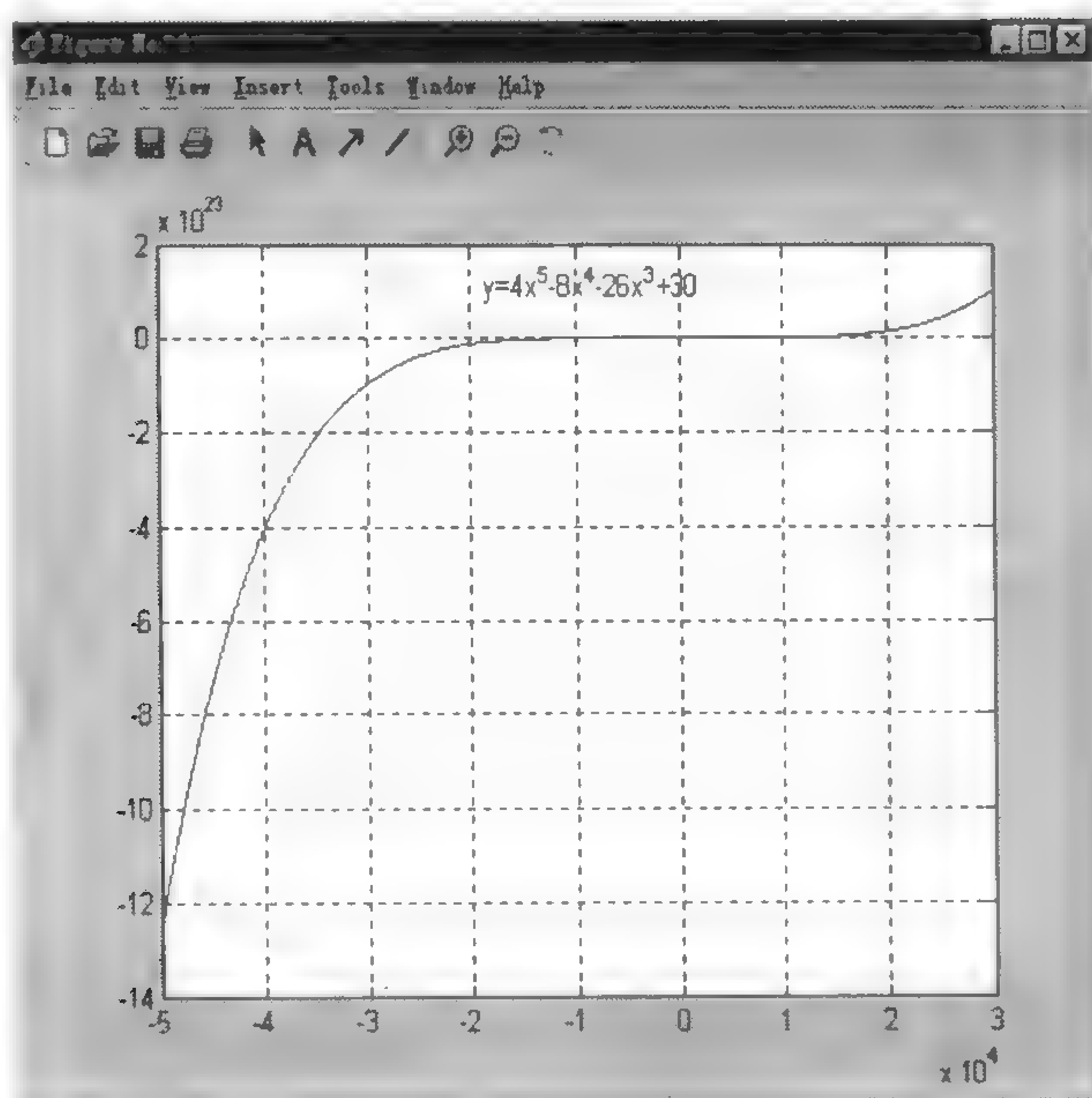

f=2xsin(x)-3
15
10
5
0
-5
-10
-15
0 1 2 3 4 5 6 7 8 9 10

(a)

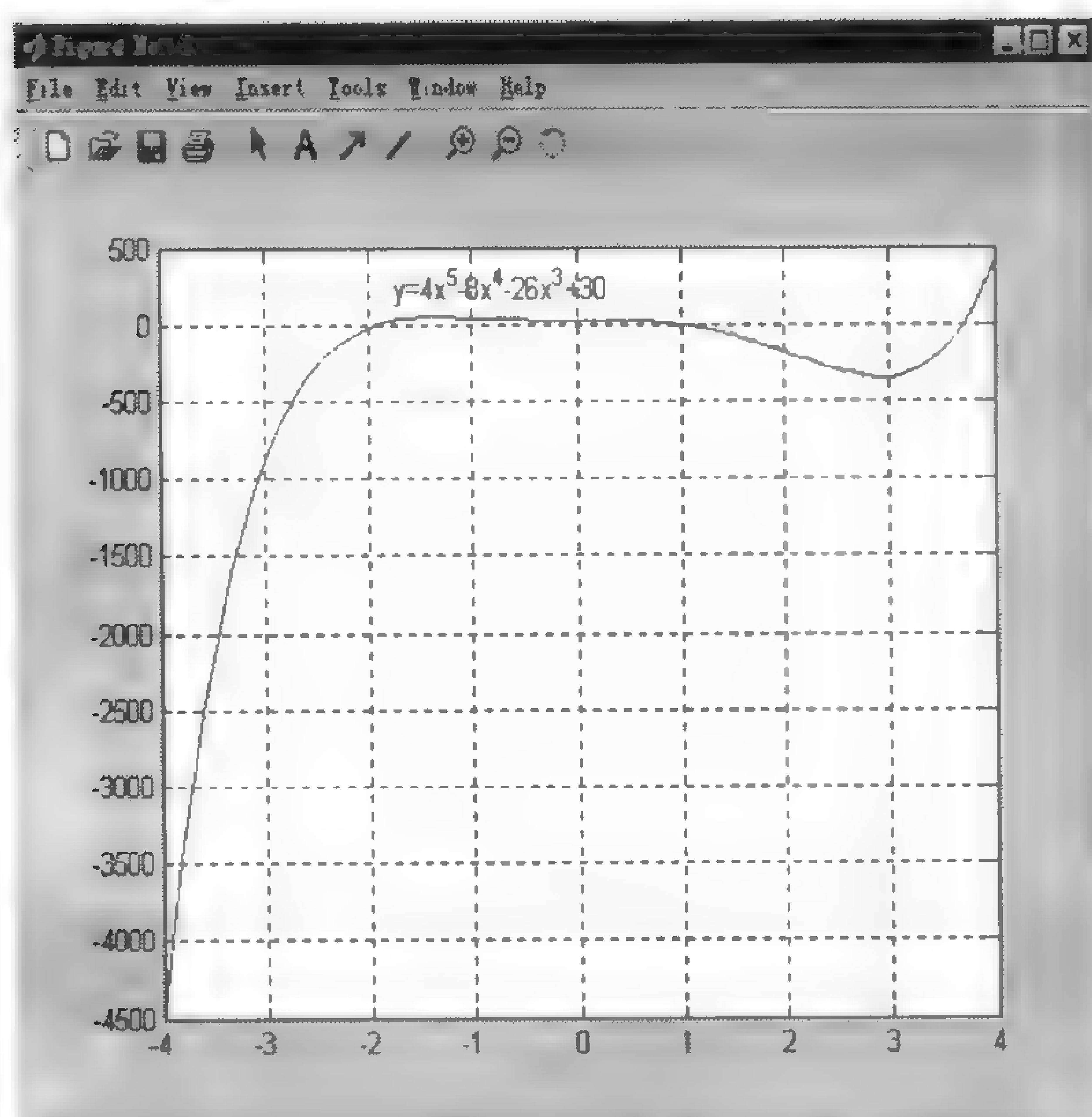

y1=2sin(x)和y2=3/x 的图形
4
3
2
1
0
-1
-2
0 1 2 3 4 5 6 7 8 9 10

(b)

图 2－1


File Edit View Insert Tools Window Help
y=4x^5-8x^4-26x^3+30
x 10^23
2
0
-2
-4
-6
-8
-10
-12
-14
-5
-4
-3
-2
-1
0
1
2
3
x 10^4

(a)


File Edit View Insert Tools Window Help
y=4x^5-8x^4-26x^3+30
500
0
-500
-1000
-1500
-2000
-2500
-3000
-3500
-4000
-4500
-4
-3
-2
-1
0
1
2
3
4

(b)

图 2 - 2

范围,画出图形即可. 例如取包含 $x=3.71$ 的区间(3.7,3.75),$h=0.01$, 得到图 2-3. 从图 2-3 中的曲线与 x 轴的交点可以看出,开区间(3.710, 3.715)内有一个实根. 同理从图 2-4 可看出(-1.950, -1.900)内有一个实根.

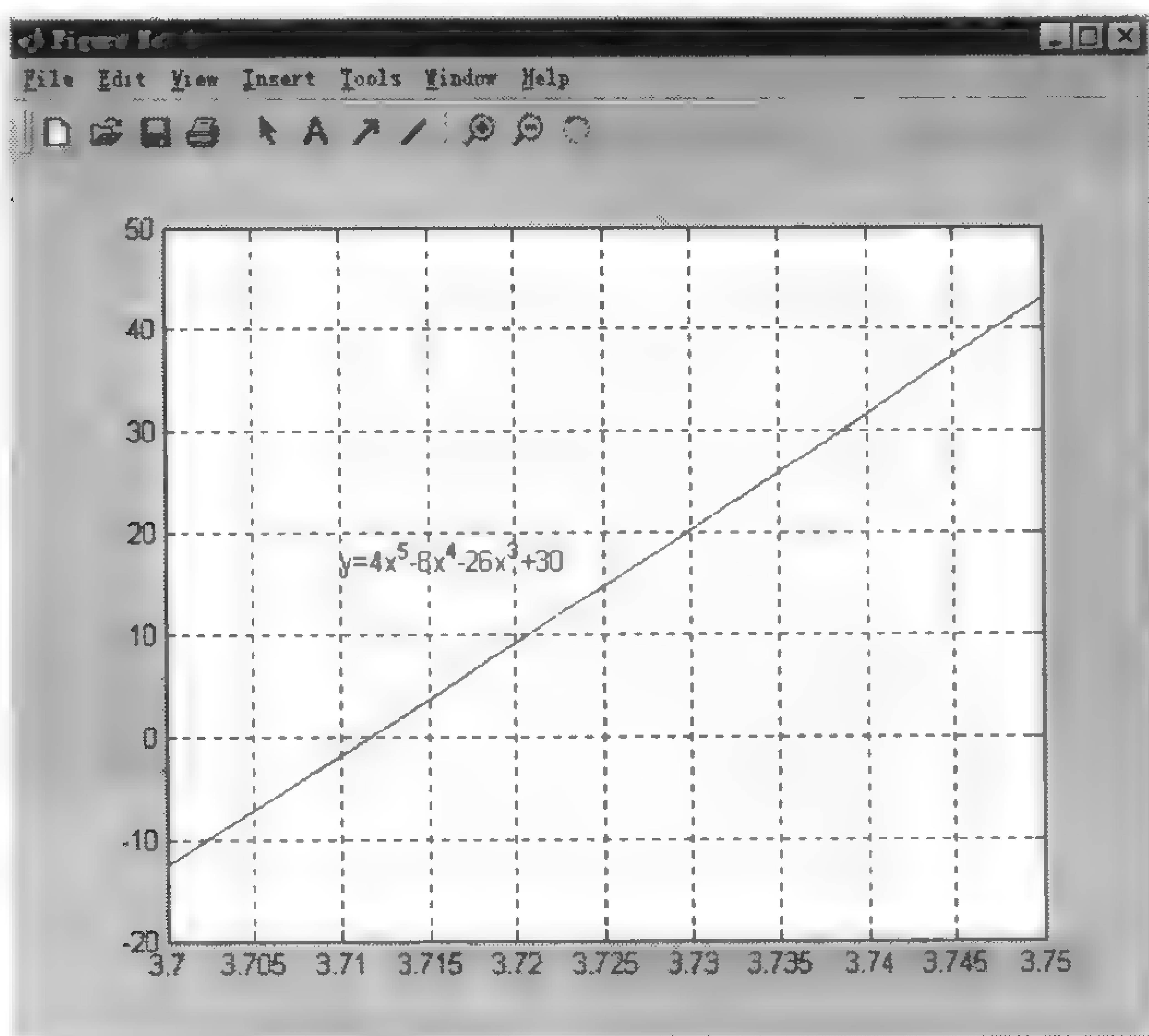


图 2-3

从例 2.2.2 可见,求解过程必须仔细,否则会丢根. 因为利用计算机绘制函数 $y=f(x)$ 在区间 $[a,b]$ 上的图形时,通常将 $[a,b]$ 划分为 $n+1$ 个等距点:

$$a=x_0<x_1<x_2<\cdots<x_n=b,$$

并计算函数值 $y_k=f(x_k)$. 然后,或者使用线段,或者利用"拟合曲线"在连续点 (x_{k-1},y_{k-1}) 和 (x_k,y_k) 之间进行绘图,其中 $k=1,2,\cdots,n$. 必须确保有足够的点,才能保证当函数变化时曲线对应的方程不丢根. 如果 $y=f(x)$ 连续,且两个邻接连续点 (x_{k-1},y_{k-1}) 和 (x_k,y_k) 位于 x 轴的两侧,则根据零点定理,在开区间 (x_{k-1},x_k) 内至少有一个根(见图 2-3 和图 2-4). 但如果在开区间 (x_{k-1},x_k) 内有一个或多个靠得很近的根,或者开区间 (x_{k-1},x_k) 与 $[a,b]$ 相比,相对很小(见图 2-2(a)中(0,1.2)),而且两个邻接连续点 (x_{k-1},y_{k-1}) 和 (x_k,y_k) 位于 x 轴的一侧或与 x 轴相交(见图 2-2(a)),则计算机产生的图形并不是函数 $y=f(x)$ 的实际图形的真实情况,这时应该采用例 2.2.2 的方法,结合函数 $y=f(x)$ 的性质和图像,适当缩小 x 的取值区间,再画函数 $y=f(x)$ 的图像(见图 2-2(b)至图 2-4),进一步研究根的分布情况.

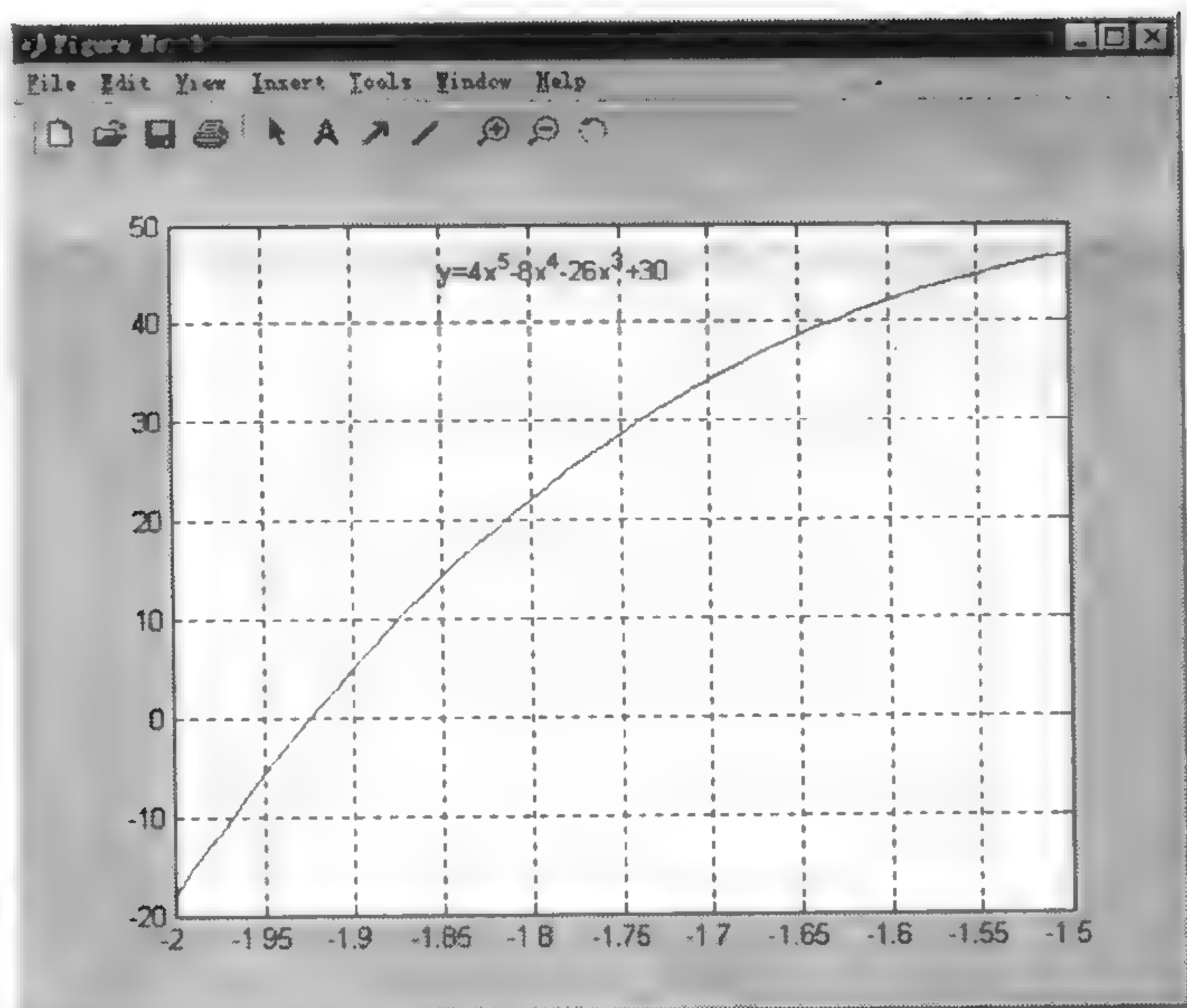


图 2-4

2.2.2 逐步搜索法及其 MATLAB 程序

逐步搜索法也称试算法. 它是求方程 $f(x)=0$ 根的近似值位置的一种常用的方法. 逐步搜索法依赖于寻找连续函数 $f(x)$ 满足 $f(a)$ 与 $f(b)$ 异号的区间 $[a,b]$. 一旦找到区间,无论区间多大,通过某种方法总会找到一个根.

(一) 逐步搜索法确定方程 $f(x)=0$ 的根 x^* 的范围的步骤

步骤 1 取含 $f(x)=0$ 根的区间 $[a,b]$,即 $f(a)\cdot f(b)<0$;

步骤 2 从 a 开始,按某个预定的步长 h (如取 $h=\dfrac{b-a}{n}$,n 为正整数),不断地向右跨步,每跨一步进行一次搜索,即检查节点 $x_k=a+kh$ 上的函数 $f(x_k)$ 值的符号,若 $f(x_{k-1})\cdot f(x_k)<0$,则可以确定一个有根区间 $[x_{k-1},x_k]$.

步骤 3 继续向右搜索,直到找出 $[a,b]$ 上的全部有根区间 $[x_{k-1},x_k]$ $(k=1,2,\cdots,n)$.

步长 h 的选取要根据函数 $f(x)$ 的性质来确定. 一般的说,对振荡剧烈,零点密集的函数,h 要选得小一些,但这要计算较多的函数值.

例 2.2.3 用逐次搜索法确定方程 $2x^3+2x^2-3x-3=0$ 根的范围.

解 设 $f(x)=2x^3+2x^2-3x-3$.

(1) 首先用 MATLAB 程序选取有根区间 $[a,b]$. 输入程序

```
>> x = -5:1:5,y = 2.*x.^3 + 2.*x.^2 - 3.*x - 3
```

运行后输出数据

```
x =   -5    -4    -3    -2    -1    0    1    2    3    4    5
y = -188   -87   -30    -5     0   -3   -2   15   60  145  282
```

从上面的数据可知 $x=-1$ 是此方程的一个根. 区间[1,2]满足 $f(1)\cdot f(2)<0$,所以(1,2)内至少有一个根. 因为 $f(x)=0$ 是三次方程,已知其有两个实根,所以它必还有另一个实根. 分析上面的数据可以看出,当 x 从 -2 增加到 0 时,$f(x)$ 的最大值至少是 0,而且很可能大于 0. 所以 $(-2,0)$ 内很可能有一个实根.

(2) 在 $(-2,0)$ 内取 $h=0.4$,输入程序

```
>> x = -2:0.4:0,y = 2.*x.^3 +2.*x.^2 -3.*x -3
```

运行后输出数据

```
x = -2.0000   -1.6000   -1.2000   -0.8000   -0.4000         0
y = -5.0000   -1.2720    0.0240   -0.3440   -1.6080   -3.0000
```

由这些函数值可以看出 $f(-1.6)\cdot f(-1.2)<0$,所以包含另一个实根的区间是 $[-1.6,-1.2]$.

从例 2.2.3 可见,在具体运用上述方法时,步长 h 的选取是关键. 很明显,只要步长 h 取得足够小,利用这种方法可以得到具有任意精度的近似根. 不过当 h 缩小时,所要搜索的步数相应增多,从而使计算量增大. 因此,我们有必要研究一种机械化算法.

(二) 收敛判定准则

设方程 $f(x)=0$ 中的函数 $y=f(x)$ 在闭区间 $[a,b]$ 上连续,开区间 (x_{k-1},x_k) 和 (x_k,x_{k+1}) 都是 $[a,b]$ 的子区间.

(1) 如果 $f(x_{k-1})\cdot f(x_k)<0$,那么这个方程在 (x_{k-1},x_k) 内至少有一个根.

(2) 如果 $f(x_{k-1})\cdot f(x_k)\geqslant 0$,而 $|f(x_k)|<\varepsilon$ (其中 ε 是给定的精度)且 $(y_{k+1}-y_k)\cdot(y_k-y_{k-1})<0$,那么 x_k 是这个方程根的近似值.

下面我们对收敛判定准则的第(2)种情形的合理性给予解释. 如果在一个区间 (x_{k-1},x_k) 内包含两个非常接近的根或一个二重根,那么很可能 $f(x_{k-1})\cdot f(x_k)\geqslant 0$,这时就不能用零点定理判别. 如果 $|f(x_k)|<\varepsilon$ (其中 ε 是给定的精度),且函数 $y=f(x)$ 在 x_k 附近有许多值接近 0,那么 x_k 可能并不接近实际的根. 因此,我们还要判别斜率在点 $(x_k,f(x_k))$ 附近变号,即 $\frac{y_k-y_{k-1}}{x_k-x_{k-1}}\cdot\frac{y_{k+1}-y_k}{x_{k+1}-x_k}<0$. 因为 $x_k-x_{k-1}>0$ 且 $x_{k+1}-x_k>0$,所以 $(y_{k+1}-y_k)\cdot(y_k-y_{k-1})<0$. 此时 x_k 是这个方程根的近似值. 因为一个序列的敛散性与产生序列的初始值和函数有关,所以不能保证将 x_k 作为初始值将一定产生一个收敛序列. 如果 x_k 是函数 $y=f(x)$ 的一个极值点,且 $|f(x_k)|<\varepsilon$ (其中 ε 是给定的精度),尽管 x_k 并不趋近于一个

根,但仍将 x_k 作为根的近似值.

(三) 逐步搜索法的 MATLAB 主程序

MATLAB 的库函数中没有逐步搜索法的程序,根据逐步搜索法的计算步骤和它的收敛判定准则编写其主程序,命名为 zhubuss.m.

逐步搜索法的 MATLAB 主程序

输入区间端点 a 和 b 的值,步长 h 和精度 tol,运行后输出迭代次数 $k=(b-a)/h+1$,方程 $f(x)=0$ 根的近似值 r.

```
function [k,r] - zhubuss(a,b,h,tol)
% 输入的量 - - - a 和 b 是闭区间[a,b]的左、右端点;
% - - -h 是步长;
% - - -tol 是预先给定的精度.
% 运行后输出的量 - - -k 是搜索点的个数;
%  - - - r 是方程 在[a,b]上的实根的近似值,其精度是 tol;
X = a:h:b;Y = funs(X);n = (b - a) /h +1;m =0;
X(n +1) =X(n);Y(n +1) = Y(n);
for k =2:n
    X(k) =a +k * h;Y(k) = funs(X(k)); % 程序中调用的 funs.m 为函数
sk = Y(k) * Y(k -1);
 if sk < =0,
     m =m +1;r(m) = X(k);
 end
 xielv = (Y(k +1) - Y(k)) * (Y(k) - Y(k -1));
 if (abs(Y(k)) < tol)&( xielv < =0)
    m =m +1;r(m) = X(k);
  end
end
```

(四) 用逐步搜索法的 MATLAB 程序求方程 $f(x)=0$ 在 $[a,b]$ 上精度是 ε 的根的步骤

步骤 1 将逐步搜索法的 MATLAB 主程序保存名为 zhubuss.m 的 M 文件.

步骤 2 建立 M 文件 funs.m

```
function    y = funs(x)
        y =f(x);
```

步骤 3 在 MATLAB 工作窗口输入如下程序

```
>>[k,r] = zhubuss(a,b,h,tol)
```

步骤 4 运行后输出结果.

例 2.2.4 用逐步搜索法的 MATLAB 程序分别求方程 $2x^3+2x^2-3x-3=0$

和 $\sin(\cos 2x^3)=0$ 在区间[-2,2]上的根的近似值,要求精度是0.000 1.

解 (1) 将逐步搜索法的 MATLAB 程序保存名为 zhubuss.m 的 M 文件.

(2) 建立 M 文件 funs.m

```
function y = funs(x)
             y = 2.*x.^3 +2.*x.^2 -3.*x -3;
```

(3) 在 MATLAB 工作窗口输入如下程序

```
>> [k,r] = zhubuss( -2,2,0.001,0.0001)
```

(4) 运行后输出的结果

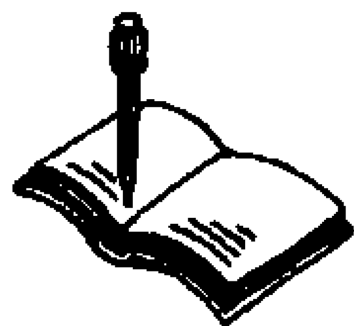
```
k  =4001
r  =  -1.2240     -1.0000     -1.0000     -0.9990     1.2250
```

即搜索点的个数为 $k=4\,001$,其中有 5 个是方程 $2x^3+2x^2-3x-3=0$ 的近似根,即 $r=-1.224\,0,-1.000\,0,-1.000\,0,-0.999\,0,1.225\,0$,其精度为 0.000 1.

在程序中将 y =2.*x.^3 +2.*x.^2 -3.*x -3 用 y = sin(cos(2.*x.^3)) 代替,可得到方程 $\sin(\cos 2x^3)=0$ 在区间[-2,2]上的根的近似值如下

```
r =     -1.9190     -1.7640     -1.5770     -1.3300     -0.9220
 0.9230    1.3310    1.5780    1.7650    1.9200
```

如果读者分别将方程 $2x^3+2x^2-3x-3=0$ 的结果与例 2.2.3 比较,方程 $\sin(\cos 2x^3)=0$的结果与例 2.1.2 比较,将会发现逐步搜索法的 MATLAB 程序的优点. 如果精度要求比较高,用这种逐步搜索法是不合算的.



习 题 2.2

1. 判别下列方程有几个实根,并求出其有根区间.

(1) $x^5-5x-3=0$;(2) $e^{-x}+x=2$;(3) $\sin 7x-3\cos 2x=0.31$.

2. 利用作图法判断下列方程在给定的区间上是否有实根,并确定实根的近似位置.

(1) $x^5-8x^4-6x^2+1=0, x\in[-15,15]$;(2) $4x^2\sin x-3=0, x\in[-10,10]$.

3. 用逐步搜索法确定下列方程在给定的区间上的根的近似值,要求精度为 0.000 1.

(1) $(x-3)^2-2\ln x=0, x\in[1,5]$;(2) $(x^2+1)^{-1}+\cos x=0, x\in[-2,2]$.

2.3 二分法及其 MATLAB 程序

2.3.1 二分法

二分法也称**逐次分半法**. 它的基本思想是:先确定方程 $f(x)=0$ 含根的区间

(a,b),再把区间逐次二等分.

(一) 判别有根区间

设函数$f(x)$在区间$[a,b]$上连续,且$f(a)\cdot f(b)<0$,那么根据零点定理知方程$f(x)=0$在开区间(a,b)内至少有一个实根.同时若$f(x)$在$[a,b]$上是单调的,那么方程$f(x)=0$在开区间(a,b)内有唯一一个实根.这是使用二分法的前提.

(二) 用二分法求方程$f(x)=0$的根x^*的近似值x_k的步骤

步骤1 若对于$a<b$,有$f(a)\cdot f(b)<0$,则在(a,b)内$f(x)=0$至少有一个根.

步骤2 取(a,b)的中点$x_1=\dfrac{a+b}{2}$,计算$f(x_1)$.

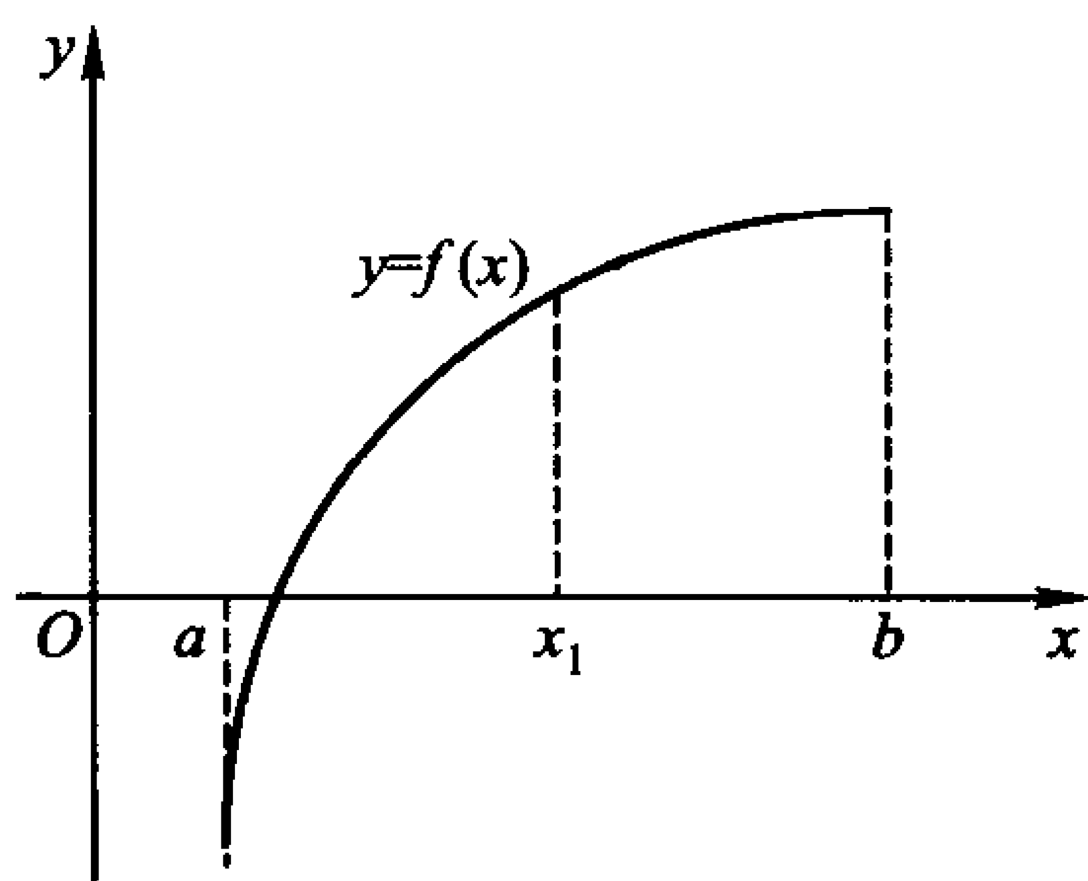


图 2-5

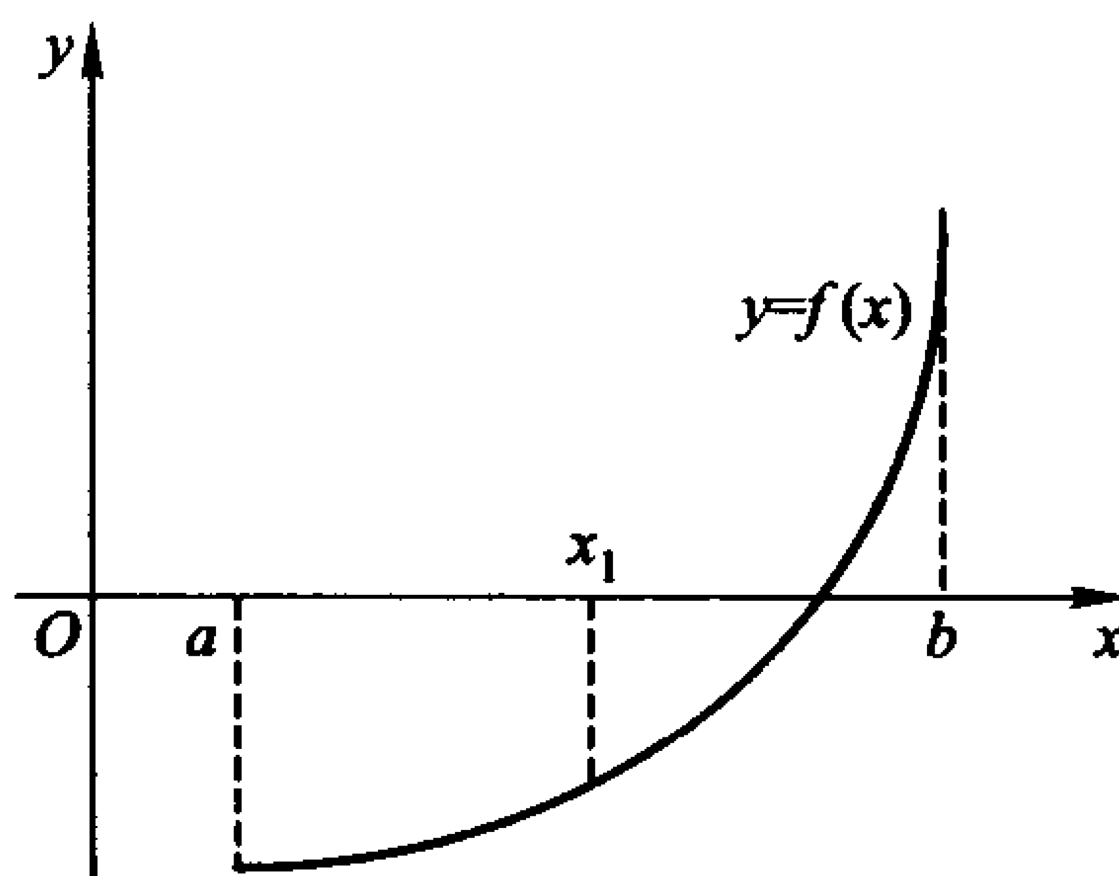


图 2-6

步骤3 若$f(x_1)=0$,则x_1是$f(x)=0$的根,停止计算,运行后输出结果$x^*=x_1$;

若$f(a)f(x_1)<0$,则在(a,x_1)内$f(x)=0$至少有一个根.取$a_1=a,b_1=x_1$(见图2-5);若$f(a)f(x_1)>0$,则取$a_1=x_1,b_1=b$(见图2-6).

步骤4 若$\dfrac{1}{2}|b_k-a_k|\leqslant\varepsilon$($\varepsilon$为预先给定的要求精度),退出计算,运行后输出结果$x^*\approx\dfrac{a_k+b_k}{2}$;反之,返回步骤1,重复步骤1,步骤2和步骤3.

(三) 二分法的收敛性

以上方法可得到每次缩小一半的区间序列$\{[a_k,b_k]\}$,(a_k,b_k)中总有方程的根.当区间长b_k-a_k很小时,取其中点$x_k=\dfrac{a_k+b_k}{2}$为根的近似值,显然有

$$|x^*-x_k|\leqslant\frac{1}{2}(b_k-a_k)=\frac{1}{2^{k+1}}(b-a)\quad(k=1,2,\cdots),\tag{2.3a}$$

由式(2.3a)可知,总有$\lim\limits_{k\to\infty}x_k=x^*$,且近似根$x^*$可以达到任意精度.

事实上,对于给定的精度要求 ε,要使

$$|x^* - x_k| \leqslant \frac{1}{2^{k+1}}(b-a) \leqslant \varepsilon,$$

只要按次数

$$k \geqslant -1 + \frac{\ln(b-a) - \ln \varepsilon}{\ln 2} \tag{2.3b}$$

逐次二分即可.

(四) 求迭代次数的 MATLAB 命令

我们可以根据式(2.3b)、区间$[a,b]$和误差 ε,编写二分法求方程根的迭代次数的 MATLAB 命令.

> **已知闭区间$[a,b]$和误差 ε. 用二分法求方程误差不大于 ε 的根的迭代次数 k 的 MATLAB 命令**
>
> ```
> k = -1 + ceil((log(b - a) - log(abtol))/log(2)) %
> ```
> ceil 是向 $+\infty$ 方向取整,abtol 是误差 ε.

例 2.3.1 证明方程 $2-3x-\sin x=0$ 在$(0,1)$内有且只有一个实根,使用二分法求误差不大于 0.000 5 的根的迭代次数.

解 设$f(x)=2-3x-\sin x$,则$f(x)$在闭区间$[0,1]$上连续,用 MATLAB 程序

```
>>x=0:1;f=2-3*x-sin(x)
```

求得$f(1)=-1.8415<0$, $f(0)=2>0$,由零点定理知,$f(x)$在$(0,1)$内至少有一个实根. 因为,$f'(x)=-3-\cos x<0, x\in(0,1)$,所以,$f(x)$在$(0,1)$内严格单调减少,故$f(x)$在$(0,1)$内至多有一个实根. 因此,已知方程在$(0,1)$内有且只有一个实根.

因为二分法求方程误差不大于 ε 的根的迭代次数的公式为

$$k \geqslant -1 + \frac{\ln(b-a) - \ln \varepsilon}{\ln 2},$$

则在 MATLAB 工作窗口输入程序

```
>>k=-1+ceil((log(1-0)-log(0.0005))/log(2))
```

得 $k=10$. 由于误差的传播,为了确保计算结果达到要求的精度,实际多取一次,需要迭代 11 次即可.

例 2.3.2 证明:方程$f(x)=x^3+4x^2-10=0$ 在区间$[1,2]$内有唯一实根. 用二分法求其根的近似值,估计使误差不超过 10^{-5}时所需要的二分次数,然后求其根的近似值.

证明 因为函数$f(x)=x^3+4x^2-10$ 在区间$[1,2]$上连续,且

$$f(1)=1^3+4\cdot1^2-10<0, f(2)=2^3+4\cdot2^2-10>0,$$

根据零点定理知方程$f(x)=0$ 在开区间$(1,2)$内至少有一个实根. 又因为在区

间[1,2]上,$f'(x)=3x^2+8x>0$,所以$f(x)$在[1,2]上是单调的,因此,方程$f(x)=0$在开区间(1,2)内有唯一一个实根.

解 用二分法求其根的近似值,用$k\geqslant -1+\dfrac{\ln(b-a)-\ln\varepsilon}{\ln 2}$估计为使误差不超过$\varepsilon=10^{-5}$时所需要的二分次数. 因为$a=1,b=2$,

$$k\geqslant -1+\frac{\ln(2-1)-\ln\varepsilon}{\ln 2}=-1+5\frac{\ln 10}{\ln 2}\approx 15.6$$

用程序

```
>> b=2;a=1;abtol=10^(-5);
 k=-1+ceil((log(b-a)-log(abtol))/log(2))
```

计算得$k=16$. 按二分法,取误差不超过10^{-5}计算,计算结果如表2-2所示.

表2-2

k	a_k	b_k	x_k	$f(x_k)$
0	1	2	1.5	2.375 000
1	1	1.5	1.25	-1.796 875
2	1.25	1.5	1.375	0.162 109
3	1.25	1.375	1.312 5	-0.848 389
4	1.312 5	1.375	1.343 75	-0.350 983
5	1.343 75	1.375	1.359 375	-0.096 409
6	1.359 375	1.375	1.367 188	0.032 356
7	1.359 375	1.367 188	1.363 281	-0.032 150
8	1.363 281	1.367 188	1.365 234	0.000 072
9	1.363 281	1.365 234	1.364 258	-0.016 047
10	1.364 258	1.365 234	1.364 746	-0.007 989
11	1.364 746	1.365 234	1.364 990	-0.003 959
12	1.364 990	1.365 234	1.365 112	-0.001 944
13	1.365 112	1.365 234	1.365 173	-0.000 936
14	1.365 173	1.365 234	1.365 204	-0.000 432
15	1.365 204	1.365 234	1.365 219	-0.000 180
16	1.365 219	1.365 234	1.365 227	-0.000 054

计算到$k=16$时,$x_{16}-a_{16}=1.365\,227-1.365\,219=8\times10^{-6}<10^{-5}$,已满足精度

要求，因此，所求的近似根为 $x_{16}=1.36523$.

值得注意的是：当 $k=16$ 时，$|f(x_{16})|=0.000054>10^{-5}$，没有达到精度的要求，这是由于误差的传播造成的. 为了确保计算结果达到要求的精度，实际计算时可以多取一次，迭代 17 次即可.

2.3.2　二分法的 MATLAB 程序

(一) 二分法的 MATLAB 主程序

二分法需自行编制程序，根据用二分法求方程 $f(x)=0$ 的根 x^* 的近似值 x_k 的步骤和式(2.3a)编写一个名为 erfen.m 的二分法的 MATLAB 主程序如下：

二分法的 MATLAB 主程序

求解方程 $f(x)=0$ 在开区间 (a,b) 内的一个根的前提条件是 $f(x)$ 在闭区间 $[a,b]$ 上连续，且 $f(a)\cdot f(b)<0$.

输入的量：a 和 b 是闭区间 $[a,b]$ 的左、右端点，*abtol* 是预先给定的精度.

运行后输出的量：k 是使用二分法的次数. x 是方程在 (a,b) 内的实根 x^* 的近似值，其精度是 *abtol*. *wuca* $=|b_k-a_k|/2$ 是使用 k 次二分法所得到的小区间的长度的一半，即实根 x^* 的近似值 x 的绝对精度限，满足 *wuca* $\leqslant$ *abtol*. $y_x=f(x_k)$，即方程 $f(x)=0$ 在实根 x^* 的近似值 x 处的函数值.

```
function [k,x,wuca,yx] = erfen(a,b,abtol)
a(1) = a; b(1) = b;
ya = fun(a(1)); yb = fun(b(1)); % 程序中调用的 fun.m 为函数
if ya * yb > 0,
disp('注意:ya * yb > 0,请重新调整区间端点 a 和 b.'), return
end
  max1 = -1 + ceil((log(b - a) - log(abtol))/log(2)); % ceil 是向
+∞ 方向取整
for k = 1: max1 + 1
     a;ya = fun(a); b;yb = fun(b); x = (a + b)/2;
     yx = fun(x); wuca = abs(b - a)/2; k = k - 1;
     [k,a,b,x,wuca,ya,yb,yx]
  if yx = = 0
     a = x; b = x;
  elseif yb * yx > 0
     b = x;yb = yx;
  else
     a - x; ya = yx;
```

```
  end
if b-a< abtol , return, end
end
k=max1; x; wuca; yx=fun(x);
```

(二) 使用二分法的主程序求解方程 $f(x)=0$ 在开区间 (a,b) 内的一个近似根的值的步骤

步骤 1 建立名为 fun.m 的 M 文件如下

```
function y1=fun(x)
          y1=f(x);
```

步骤 2 将二分法的主程序保存名为 erfen.m 的 M 文件;

步骤 3 在 MATLAB 工作窗口输入程序

```
>> [k,x,wuca,yx]=erfen (a,b, abtol)
```

其中输入的量:区间端点的值 a,b 和精度是 $abtol$ 都是具体给定的数值,然后按运行键.运行后输出计算次数 k、使用 k 次二分法所得到的小区间 $[a_k,b_k]$ 的中点的值 x 和它的函数值 $y(x)$ 及 $wuca=|b_k-a_k|/2$(参见例 2.3.3).

例 2.3.3 确定方程 $x^3-x+4=0$ 的实根的分布情况,并用二分法求在开区间 $(-2,-1)$ 内的实根的近似值,要求精度为 0.001.

解 (1)先用两种方法确定方程 $x^3-x+4=0$ 的实根的分布情况.

方法 1 作图法.

在 MATLAB 工作窗口输入程序

```
>>x=-4:0.1:4;
y=x.^3-x +4; plot(x,y)
grid,gtext('y=x.^3-x+4')
```

画出函数 $f(x)=x^3-x+4$ 的图像,如图 2-7.从图像可以看出,此曲线有两个驻点 $\pm\frac{\sqrt{3}}{3}$ 都在 x 轴的上方,在 $(-2,-1)$ 内曲线与 x 轴只有一个交点,则该方程有唯一一个实根,且在 $(-2,-1)$ 内.

方法 2 试算法.

在 MATLAB 工作窗口输入程序

```
>>x=-4:1:4,y=x.^3-x +4
```

运行后输出结果

```
x =    -4    -3    -2    -1    0    1    2    3    4
y =   -56   -20    -2     4    4    4   10   28   64
```

由于连续函数 $f(x)$ 满足 $f(-2)\cdot f(-1)<0$,所以此方程在 $(-2,-1)$ 内有一个实根.

(2) 用两种方法求方程在 $(-2,-1)$ 内的一个实根.

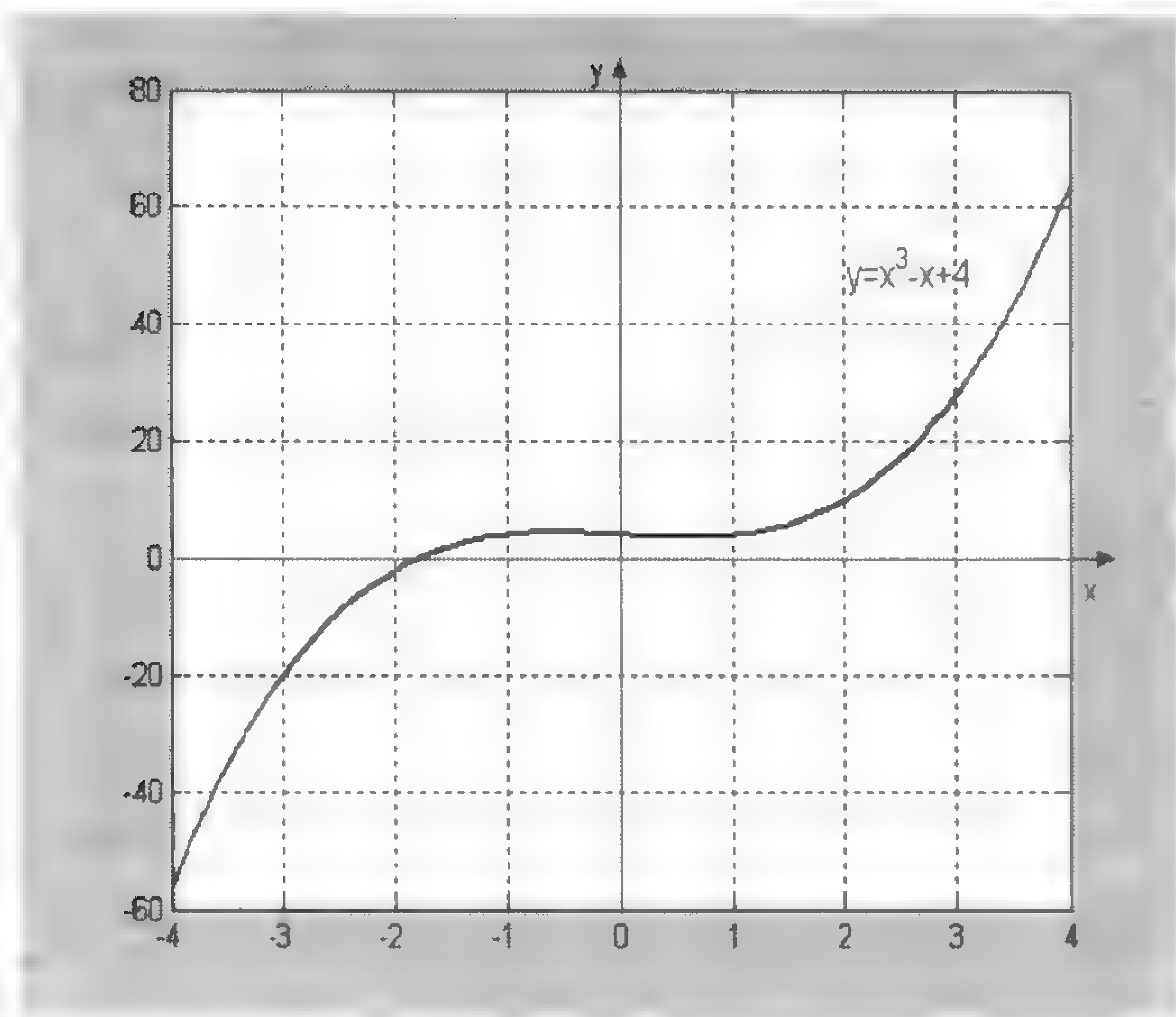


图 2-7

方法 1 利用二分法逐次寻找方程的根.

① 先求二分区间的次数 k

由 $k > -1 + \dfrac{\ln(b-a) - \ln \varepsilon}{\ln 2} = -1 + \dfrac{\ln 1 - \ln 0.001}{\ln 2} \approx 8.9668$,也可以用程序

```
>> b = -1;a = -2;abtol =0.001;
   k = -1 +ceil((log(b-a) - log(abtol))/log(2)) % ceil 是 +∞ 方向
```

取整.

运行后输出结果 $k=9$. 只要按次数 $k=9$ 逐次二分即可得到绝对精度为 0.001 的实根的近似值 x_9.

② 求方程在 $(-2,-1)$ 内有一个实根的近似值 x_9.

取 $[-2,-1]$ 的中点 $x_1 = \dfrac{a_1+b_1}{2} = -1.5$,计算 $f(x_1)=2.125$,由于 $f(-2)\cdot f(-1.5)<0$,则表明根位于区间 $(-2,-1.5)$ 内,所以取 $b_2 = x_1 = -1.5$,$a_2 = a_1 = -2$. 按这样的方法,可得到序列 $\{x_k\}$. 将这种利用二分法计算的结果列入表 2-3. 因为 $\left|\dfrac{b_9-a_9}{2}\right| \approx 0.001$,取 $x \approx x_9 \approx -1.796$.

表 2-3

次数 k	左端点 a_k	右端点 b_k	中点 x_k	$\left\|\frac{b_k-a_k}{2}\right\|$	函数值 $f(a_k)$	函数值 $f(b_k)$	函数值 $f(x_k)$
0	-2.000 0	-1.000 0	-1.500 0	0.500 0	-2.000 0	4.000 0	2.125 0
1	-2.000 0	-1.500 0	-1.750 0	0.250 0	-2.000 0	2.125 0	0.390 6
2	-2.000 0	-1.750 0	-1.875 0	0.125 0	-2.000 0	0.390 6	-0.716 8
3	-1.875 0	-1.750 0	-1.812 5	0.062 5	-0.716 8	0.390 6	-0.141 8
4	-1.812 5	-1.750 0	-1.781 3	0.031 3	-0.141 8	0.390 6	0.129 6
5	-1.812 5	-1.781 3	-1.796 9	0.015 6	-0.141 8	0.129 6	-0.004 8
6	-1.796 9	-1.781 3	-1.789 1	0.007 8	-0.004 8	0.129 6	0.062 7
7	-1.796 9	-1.789 1	-1.793 0	0.003 9	-0.004 8	0.062 7	0.029 0
8	-1.796 9	-1.793 0	-1.794 9	0.002 0	-0.004 8	0.029 0	0.012 1
9	-1.796 9	-1.794 9	-1.795 9	0.001 0	-0.004 8	0.012 1	0.003 7

方法 2 用二分法的主程序计算.

① 建立名为 fun.m 的 M 文件

```
function y1 = fun(x)
        y1 = x^3 - x + 4;
```

② 将二分法的主程序保存名为 erfen.m 的 M 文件.

③ 在 MATLAB 工作窗口输入程序

```
>> [k,x,wuca,yx] = erfen ( -2, -1,0.001)
```

④ 运行后屏幕显示用二分法计算过程被列入表 2-3,其余结果为

```
k = 9,x = -1.7959,
wuca = 9.7656e-004,yx = 0.0037
```

注意 用二分法的主程序 erfen.m 求方程 $f(x)=0$ 在闭区间 $[a,b]$ 上的近似根时,如果输出结果为"注意:ya * yb >0,请重新调整区间端点 a 和 b.",则该方程在开区间 (a,b) 内可能无实根,也可能有实根. 例如,方程 $x^3-x+4=0$ 在(0,1)内无实根,在 MATLAB 工作窗口输入程序

```
>> [k, x,wuca,yx] = erfen (0,1,0.005)
```

则运行结果为

注意:ya * yb >0,请重新调整区间端点 a 和 b.

再如,方程 $x^2-2x+4=0$ 在(1,3)内有二重实根 2,重新建立名为 fun.m 的 M 文件

```
function y1 = fun(x)
         y1 = x^2 - 2 * x + 4;
```

在 MATLAB 工作窗口输入程序

```
>> [k, x,wuca,yx] = erfen (1,3,0.005)
```

则运行结果与前例相同。

可见，**二分法的优点是对函数的要求低（只要求 $f(x)$ 满足零点定理的条件），方法简便、可靠、程序设计容易，事先估计计算次数容易，收敛速度恒定；缺点是不能求出方程的偶重根，收敛速度较慢.**

例如，运行程序

```
>> x = 0:0.1:3; y = x.^3 - 5.* x.^2 + 8.* x - 4;
plot(x,y)
grid,gtext('y = x^3 - 5x^2 + 8x - 4')
```

得到图 2 - 8，可见，$f(x)=x^3-5x^2+8x-4$ 在单根 1 附近两边函数值异号，可以用二分法，二重根 2 附近两边函数值同号，不能用二分法求之.

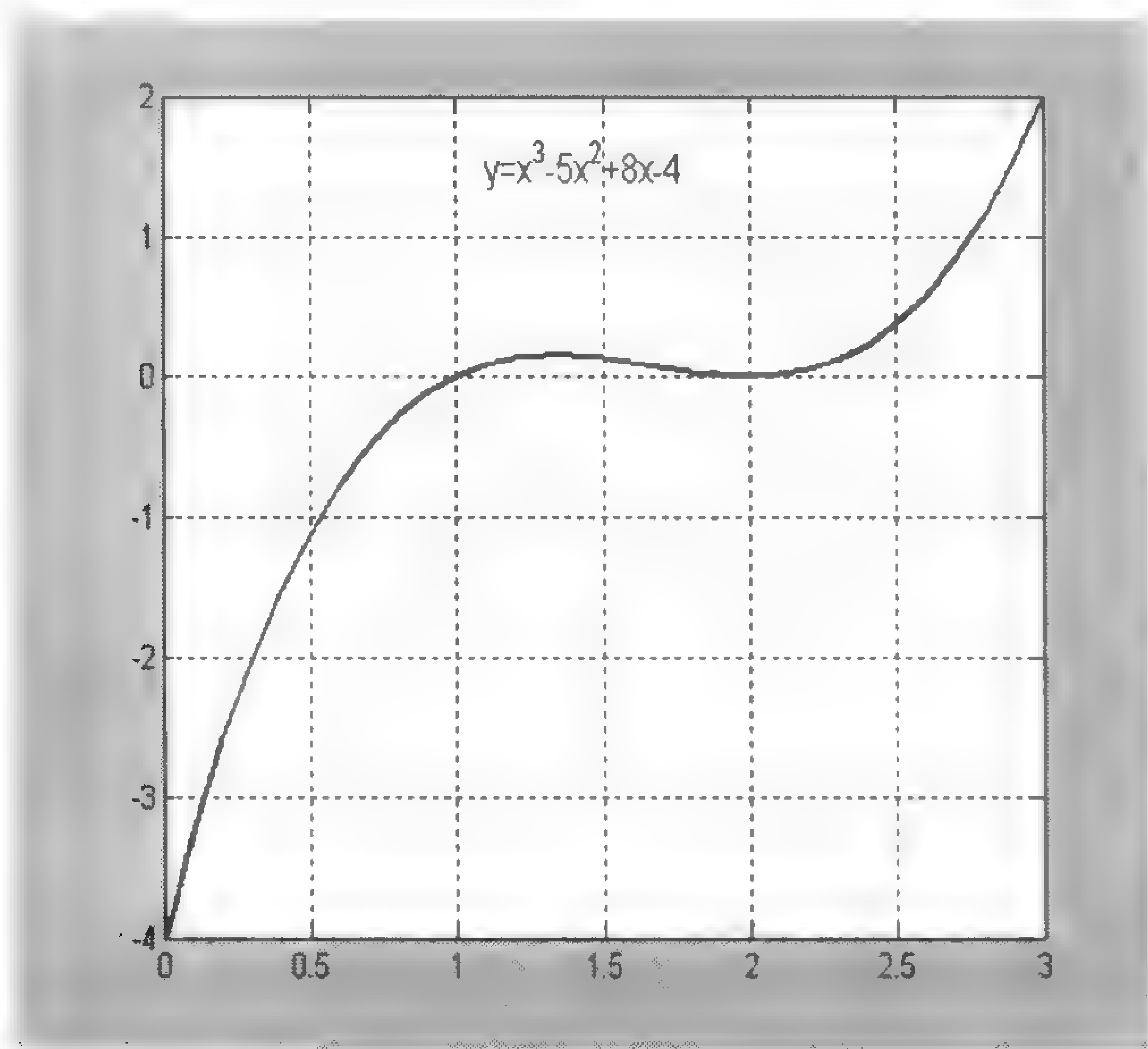


图 2 - 8

二分法的思想方法还可用于搜索一个较大区间 $[a,b]$ 内实根的分布情况（不包括偶重实根）. 实际的做法是取适当的步长 $h=\dfrac{b-a}{n}$（n 为正整数），逐一

检验小区间$[a+kh,a+(k+1)h]$ $(k=0,1,2,\cdots,n-1)$的两端函数值. 若异号，则按以上二分法求出其中的根；若同号，则不作求根，而转入检查下一个区间，只要h选得较小，则可求出本区间内的所有奇重根(包括单实根). h选得过大，可能漏掉某些根；h选得过小，则计算量增大.

例 2.3.4 已知函数$g(x)=x\sin 2x$. 用二分法迭代8次，寻找在闭区间$[1.5,2.5]$上的值x，满足$g(x)=-1.5$（函数$x\sin 2x$用弧度计算），且求其绝对误差和相对误差.

解 (1) 用二分法寻找$f(x)=x\sin 2x+1.5$在$[1.5,2.5]$上的零点，首先用画图法判断在给定的区间上此函数有几个零点.

在 MATLAB 工作窗口输入程序

```
>> x =1.5:0.001:2.5;f =x.* sin(2.*x) +1.5;
 plot(x,f),grid,gtext('f =x.* sin(2.*x) +1.5')
```

运行后，画出图2-9，可见函数$f(x)$在$[1.5,2.5]$上有唯一一个零点，且在2附近.

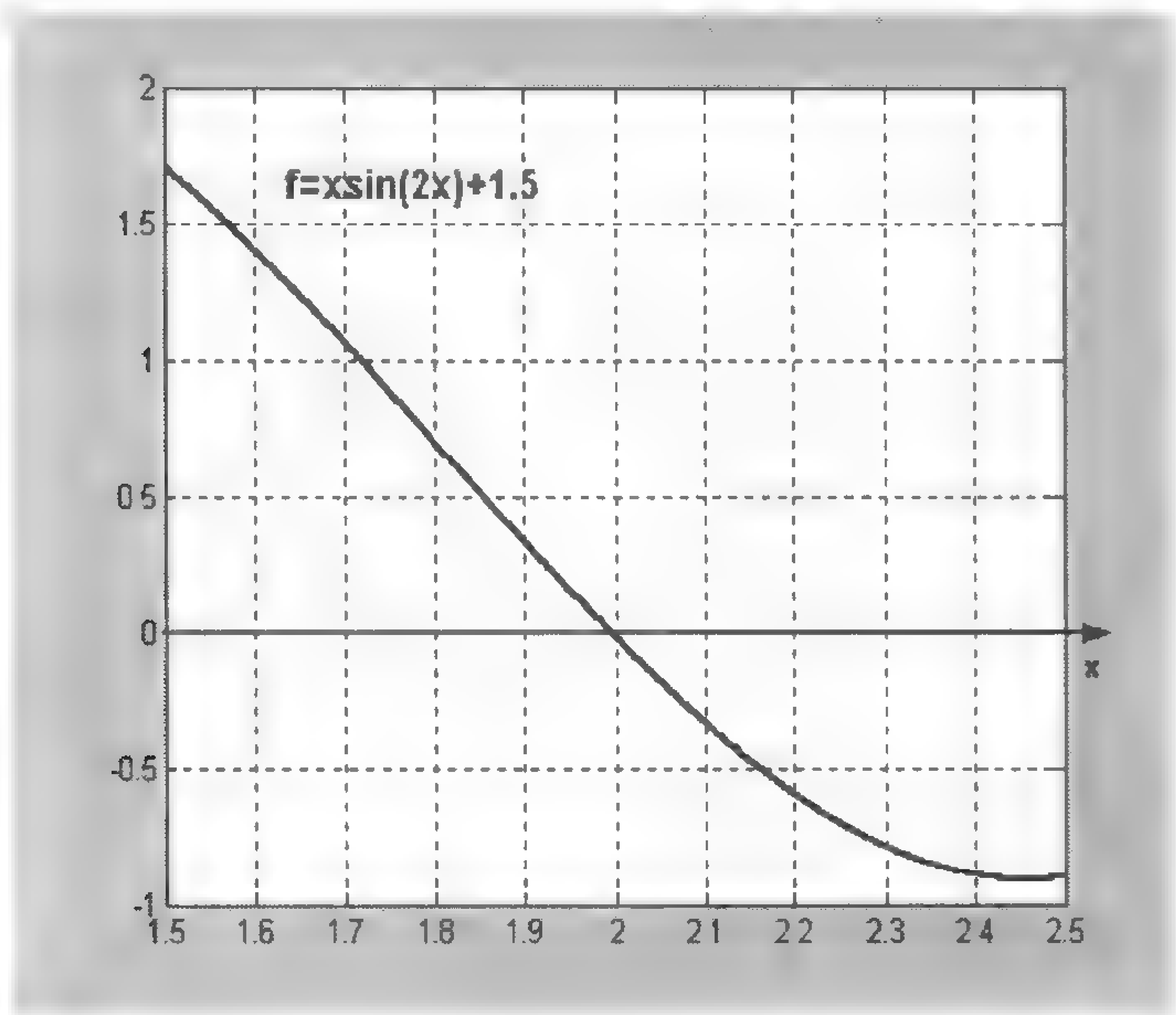


图 2-9

(2) 根据二次迭代法的绝对误差估计公式

$$|x_k-x^*|\leqslant\frac{1}{2^{k+1}}(b-a)\leqslant\varepsilon,$$

取 $k=8, a=1.5, b=2.5$. 在 MATLAB 工作区输入程序

```
>> k=8;a=1.5;b=2.5; abtol=(b-a)/(2^(k+1))
```

运行后得绝对误差为

```
abtol =
    0.00195312500000
```

(3) 建立名为 fun.m 的 M 文件

```
function y1 = fun(x)
          y1 = x.*sin(2.*x)+1.5;
```

(4) 在 MATLAB 工作窗口输入程序

```
>> [k,xk,wuca,yx] = erfen(1.5,2.5, 0.001953125)
```

运行得迭代次数 $k=8$;所要求的近似根 $x_k=1.994\,140\,625$, $wuca\ =0.001\,953\,125$,其函数值 $y_x=0.006\,207\,565$.

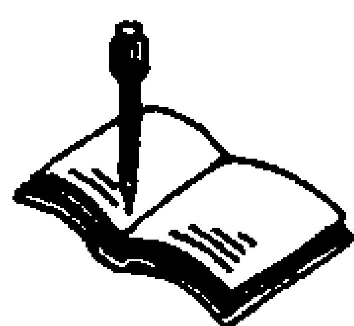
(5) 根据相对误差公式

$$\frac{|x_k-x^*|}{|x_k|}\leqslant\frac{\varepsilon}{|x_k|}=\delta,$$

编写 MATLAB 程序

```
>>abtol =0.001953125; xk =1.994140625; tol = abtol/abs(xk)
```

运行得 x_k 的相对误差 $tol\ =\ 9.794\,319\,295e-004$.



习 题 2.3

1. 用二分法确定下列方程在给定的区间上的根的近似值,要求精度为 0.000 5.

(1) $2x-3\tan x=0, x\in[-2,-0.5]$;(2) $3x^2-e^x=0,\ x\in[-1,0]$.

2. 证明方程 $1-2x-\sin x=0$ 在(0,1)内有且只有一个实根,使用二分法求误差不大于 0.000 01 的根的迭代次数.

3. 在无阻尼强迫振荡的研究中会碰到函数 $f(x)=x\sin x$. 用二分法迭代 32 次,寻找在闭区间[0,2]上的值 x,满足 $f(x)=1$ (函数 $x\sin x$ 用弧度计算),且求其误差.

4. 利用二分法求 $\sqrt[3]{25}$ 的近似值,精确到两位有效数字.

5. 证明方程 $x^3-x-1=0$,在[1,2]上有且只有一个实根,求使精度达到 10^{-4} 的近似根和迭代次数.

6. 对下列方程求根,精度到 10^{-5}.

(1) $x-2^{-x}=0, 0\leqslant x\leqslant 1$;

(2) $e^x+2^{-x}+2\cos x-6=0, 1\leqslant x\leqslant 2$;

(3) $e^x-x^2+3x-2=0, 0\leqslant x\leqslant 1$.

2.4 迭代法及其 MATLAB 程序

确定根的近似位置以后,接下来的工作就是将根精确化,即按某种方法将初始近似值逐步精确化,直到满足所要求的精确度为止. 上节介绍的二分法是将根精确化的方法之一,但是它的收敛速度较慢,且不能求出偶重根. 迭代法可以克服这种缺陷. 迭代法是求解方程的根、线性和非线性方程组的解的基本而重要的方法.

2.4.1 迭代法的基本思想

迭代法的基本思想是利用某种迭代公式,使某个近似根逐步精确化,直到得到满足精度要求的近似根为止.

把给定的方程 $f(x)=0$ 改写成容易迭代的形式 $x=\varphi(x)$ (其中 $\varphi(x)$ 称为迭代函数), 选择合适的初始值 x_0,代入 $\varphi(x)$ 算得的结果记为 $x_1=\varphi(x_0)$,一般 $x_1\neq x_0$;把 x_1 代入 $\varphi(x)$ 算得的结果记为 $x_2=\varphi(x_1)$,…. 若把 x_k 代入 $\varphi(x)$,一般有迭代公式

$$x_{k+1}=\varphi(x_k)\ \ (k=0,1,2,\cdots), \tag{2.4}$$

由此得到迭序列 $x_0,x_1,\cdots,x_k,\cdots$. 若迭代序列 $\{x_k\}$ 收敛到 x^* (即 $\lim\limits_{k\to\infty}x_k=x^*$),则当函数 $\varphi(x)$ 连续时,由式(2.4)得

$$\lim_{k\to\infty}x_k=\lim_{k\to\infty}\varphi(x_k)=\varphi(\lim_{k\to\varphi}x_k).$$

$x^*=\varphi(x^*)$,称为 φ 的**不动点**,即 x^* 为原方程 $f(x)=0$ 的根. 一般计算有限步,即可得到某种精度的近似根 x_k. 这种求方程根的方法称为**简单迭代法**,或**逐次逼近法**.

当然,若 $\{x_k\}$ 发散,迭代法就失败.

2.4.2 迭代法的 MATLAB 程序 1

迭代法需要自行编制程序,这里主要介绍迭代法的 MATLAB 程序 1,并且通过实例说明这个程序的应用和它的可行性. 迭代法的 MATLAB 程序 1 使用时只需输入迭代初始值 x_0、迭代次数 k、迭代公式 $x_{k+1}=\varphi(x_k)$ 和一条命令,运行后就可以输出迭代序列 $\{x_k\}$、迭代 k 次得到的迭代值 x_k 和相邻两次迭代的偏差 *piancha* $=|x_k-x_{k-1}|$ (简称**偏差**)和偏差的相对误差 *xdpiancha* $=|x_k-x_{k-1}|/|x_k|$ 的值,并且具有警报功能(若迭代序列发散,则提示用户"请重新输入新的迭代公式";若迭代序列收敛,则屏幕会出现"祝贺您! 此迭代序列收敛,且收敛速度

较快").我们可以用这个程序来判断迭代序列的敛散性,也可以用于比较由一个方程得到的几个迭代公式的敛散性的优劣.

如何选择 $\varphi(x)$ 构造迭代公式,对于收敛速度是至关重要的.用同一个方程构成的不同的迭代公式,所得到的迭代序列 $\{x_n\}$ 的敛散性不同,且收敛的速度也不同.迭代法的 MATLAB 程序 1 主要用于由迭代公式所得到的迭代序列的敛散性的判别,也可以用这个程序来比较由一个方程得到的几个迭代公式所得到的迭代序列的收敛速度.

(一) 迭代法的 MATLAB 程序 1

> **迭代法的 MATLAB 程序 1**
>
> 输入的量:初始值 x_0、迭代次数 k 和迭代公式 $x_{k+1}=\varphi(x_k)$ $(k=0,1,2,\cdots)$;
>
> 运行后输出的量:迭代序列 $\{x_k\}$、迭代 k 次得到的迭代值 x_k、相邻两次迭代的偏差 $piancha=|x_k-x_{k-1}|$ 和它的偏差的相对误差 $xdpiancha=|x_k-x_{k-1}|/|x_k|$ 的值.

根据迭代公式(2.4)和已知条件,编写一个名为 diedai1.m 的 M 文件.

```
function [k,piancha,xdpiancha,xk] = diedai1(x0,k)
% 输入的量 - -x0 是初始值,k 是迭代次数
x(1) = x0;
for i =1:k
  x(i +1) = fun1(x(i));% 程序中调用的 fun1.m 为函数 y = φ(x)
  piancha = abs(x(i +1) -x(i)); xdpiancha = piancha/( abs(x(i +
1)) +eps);
   i =i +1;xk =x(i);[(i -1) piancha xdpiancha xk]
end
if (piancha >1)&(xdpiancha >0.5)&(k >3)
    disp('请用户注意:此迭代序列发散,请重新输入新的迭代公式')
    return;
  end
    if (piancha < 0.001)&(xdpiancha < 0.0000005)&(k >3)
      disp('祝贺您! 此迭代序列收敛,且收敛速度较快')
      return;
  end
p =[(i -1) piancha xdpiancha xk]';
```

(二) 使用迭代法的程序 1 求迭代公式 $x_{k+1}=\varphi(x_k)$ $(k=0,1,2,\cdots)$ 的值的

步骤

步骤 1 把给定的方程 $f(x)=0$ 改写成容易迭代的形式 $x=\varphi(x)$；

步骤 2 建立 M 文件 fun1.m 如下

```
function y1 = fun1(x)
         y1 = φ(x);
```

步骤 3 保存名为 diedai1.m 的 M 文件；

步骤 4 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk] = diedai1(x0,k)
```

其中输入的量:初始值 x_0,迭代次数 k 都是具体给定的数值. 运行后输出迭代次数 k、迭代序列 $\{x_k\}$、它的相邻两次迭代的偏差 $piancha=|x_k-x_{k-1}|$和它的相对误差 $xdpiancha=|x_k-x_{k-1}|/|x_k|$的值. 当 $piancha>1$, $xdpiancha>0.5$, $k>3$ 时，运行后分别输出错误信息'请用户注意:此迭代序列发散,请重新输入新的迭代公式';当 $piancha<0.001$, $xdpiancha<0.0000005$, $k>3$ 时运行后输出祝贺信息'祝贺您！此迭代序列收敛,且收敛速度较快'. 下面通过例 2.4.1 说明这个程序的应用和它的可行性.

例 2.4.1 求方程 $f(x)=x^2+2x-10$ 的一个正根.

解 (1) 构造不同的函数 $\varphi(x)$.

因为 $f(2)=-2$, $f(3)=5$, 所以有一个正根 $x^*\in(2,3)$. 当然用二次方程求根公式不难得到 $x^*=2.31662479035540$. 下面我们构造不同的迭代函数 $\varphi(x)$,再以 $x_0=2$ 为初始值,分别用下面的三种迭代法求解.

$x=\varphi_1(x)=(10-x^2)/2$,迭代公式 $x_{k+1}=(10-x_k^2)/2$;

$x=\varphi_2(x)=10/(x+2)$,迭代公式 $x_{k+1}=10/(x_k+2)$;

$x=\varphi_3(x)=x-(x^2+2x-10)/(2x+2)$,迭代公式 $x_{k+1}=x_k-(x_k^2+2x_k-10)/(2x_k+2)$.

(2) 利用迭代法的 MATLAB 程序 1 计算迭代序列 $\{x_k\}$.

① 将名为 diedai1.m 的 M 文件.

② 建立 M 文件 fun1.m

```
function y1 = fun1(x)
         y1 = (10 - x^2)/2;
```

说明 名为 fun1.m 的 M 文件的函数也可以用 $y=10/(x+2)$ 或 $y=x-(x\text{^}2+2*x-10)/(2*x+2)$ 替换.

③ 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk] = diedai1(2,5)
```

④ 用迭代公式 $x_{k+1}=(10-x_k^2)/2$ 运行后输出的结果

```
[k,piancha,xdpiancha,xk] =
1.00000000000000  1.00000000000000  0.33333333333333  3.00000000000000
2.00000000000000  2.50000000000000  5.00000000000000  0.50000000000000
3.00000000000000  4.37500000000000  0.89743589743590  4.87500000000000
4.00000000000000  11.75781250000000  1.70828603859251  -6.88281250000000
5.00000000000000  11.80374145507813  0.63167031671297  -18.68655395507813
请用户注意:此迭代序列发散,请重新输入新的迭代公式
k =5,piancha = 11.80374145507813,xdpiancha = 0.63167031671297,
xk = -18.68655395507813
```

由以上运行后输出的迭代序列与根 $x^* = 2.316\ 624\ 790\ 355\ 40$ 相差越来越大,即迭代序列 $\{x_k\}$ 发散,此迭代法就失败. 这时偏差 *piancha* 逐渐增大且偏差的相对误差 *xdpiancha* 的值大于 0.5.

⑤ 用迭代公式 $x_{k+1} = 10/\ (x_k + 2)$ 运行后输出的结果

```
[k,piancha,xdpiancha,xk] =
1.00000000000000  0.50000000000000  0.20000000000000  2.50000000000000
2.00000000000000  0.27777777777778  0.12500000000000  2.22222222222222
3.00000000000000  0.14619883040936  0.06172839506173  2.36842105263158
4.00000000000000  0.07926442612555  0.03462603878116  2.28915662650602
5.00000000000000  0.04230404765128  0.01814486863115  2.33146067415730
k =5,piancha =0.04230404765128,xdpiancha = 0.01814486863115,
xk = 2.33146067415730
```

可见,偏差 *piancha* 和偏差的偏差的相对误差 *xdpiancha* 的值逐渐变小,且第 5 次的迭代值 $x_k = 2.331\ 460\ 674\ 157\ 30$ 与根 $x^* = 2.316\ 624\ 790\ 355\ 40$ 接近,则迭代序列 $\{x_k\}$ 收敛,但收敛速度较慢,此迭代法较为成功.

⑥ 用迭代公式 $x_{k+1} = x_k - (x_k^2 + 2x_k - 10)/\ (2x_k + 2)$ 运行后输出的结果

```
[k,piancha,xdpiancha,xk] =
1.00000000000000  0.33333333333333  0.14285714285714  2.33333333333333
2.00000000000000  0.01666666666667  0.00719424460432  2.31666666666667
3.00000000000000  0.00004187604690  0.00001807631822  2.31662479061977
4.00000000000000  0.00000000026437  0.00000000011412  2.31662479035540
5.00000000000000                 0                 0  2.31662479035540
祝贺您! 此迭代序列收敛,且收敛速度较快
k = 5; piancha = 0; xdpiancha = 0; y = 2.31662479035540.
```

可见,偏差 *piancha* 和偏差的相对误差 *xdpiancha* 的值越来越小,且第 5 次的迭代值 $x_k = 2.316\ 624\ 790\ 355\ 40$ 与根 $x^* = 2.316\ 624\ 790\ 355\ 40$ 相差无几,则

迭代序列$\{x_k\}$收敛,且收敛速度很快,此迭代法成功.

⑦ 将3个迭代公式前5步的迭代序列$\{x_k\}$的计算、偏差 *piancha* 和偏差的相对误差 *xdpiancha* 值的结果列在一起(见表2-4),进行比较.

表2-4 3个迭代公式的计算结果

迭代公式	迭代第5次的偏差	迭代第5次的偏差	x_0	x_1	x_2	x_3	x_4	x_5
φ_1	11.803 7	0.631 7	2.000 0	3.000 0	0.500 0	4.875 0	-6.882 8	-18.687
φ_2	0.042 3	0.018 1	2.000 0	2.500 0	2.222 2	2.368 4	2.289 2	2.331 5
φ_3	0	0	2.000 0	2.333 3	2.316 7	2.316 6	2.316 6	2.316 6

可以看出,φ_1 根本不收敛,它的偏差 *piancha* 很大且偏差的相对误差 *xdpiancha* 的值大于0.5;φ_2 虽呈现收敛趋势,但很慢,它的偏差小于0.4且偏差的相对误差小于0.02;φ_3 收敛很快,偏差和偏差的相对误差几乎为0. 由此可见,迭代序列$\{x_k\}$的敛散性与迭代公式有关,也与相邻两次迭代的偏差和偏差的相对误差有关,它们的值越小,迭代序列$\{x_k\}$的收敛速度越快. 对于一般情形也是成立的,这可由下面2.4.4迭代序列收敛性的定理得出.

2.4.3 迭代法的几何解释

求方程 $x=\varphi(x)$ 的根,可归结为找直线 $y=x$ 与曲线 $y=\varphi(x)$ 的交点 P 的横坐标 x^*(参见图2-10至图2-13). 对于 x^* 的某个初始值 x_0,对应曲线 $y=\varphi(x)$ 上一个点 P_0,过点 P_0 引平行于 x 轴的直线,交直线 $y=x$ 于点 Q_0,过 Q_0 再作平行于 y 轴的直线,它与曲线 $y=\varphi(x)$ 交于点 P_1,容易看出,P_1 的横坐标为 $x_1=\varphi(x_0)$. 按图所示沿带箭头的路线继续做下去,就在曲线 $y=\varphi(x)$ 上得到点列 $P_0,P_1,P_2,\cdots$,其横坐标分别为迭代公式 $x_{k+1}=\varphi(x_k)$ $(k=0,1,2,\cdots)$ 求得的迭代值 $x_0,x_1,x_2,\cdots$. 如果点列 $P_0,P_1,P_2,\cdots$ 逐次逼近交点 P^*,这时迭代数列 $x_0,x_1,x_2,\cdots$ 收敛(如图2-10和图2-12). 否则,就不收敛(如图2-11和图2-13). 图形直观地告诉我们,之所以有如此截然不同的结果,是由于在 x^* 附近图2-10和图2-12中的曲线 $y=\varphi(x)$ 较平缓,而图2-11和图2-13中的 $y=\varphi(x)$ 较陡峭. 通过作图可以发现,在 x^* 附近,若曲线 $y=\varphi(x)$ 的斜率(取绝对值)小于直线 $y=x$ 的斜率($=1$),则序列$\{x_n\}$将收敛;反之,当 $y=\varphi(x)$ 的斜率(取绝对值)大于1时,序列$\{x_n\}$不收敛.

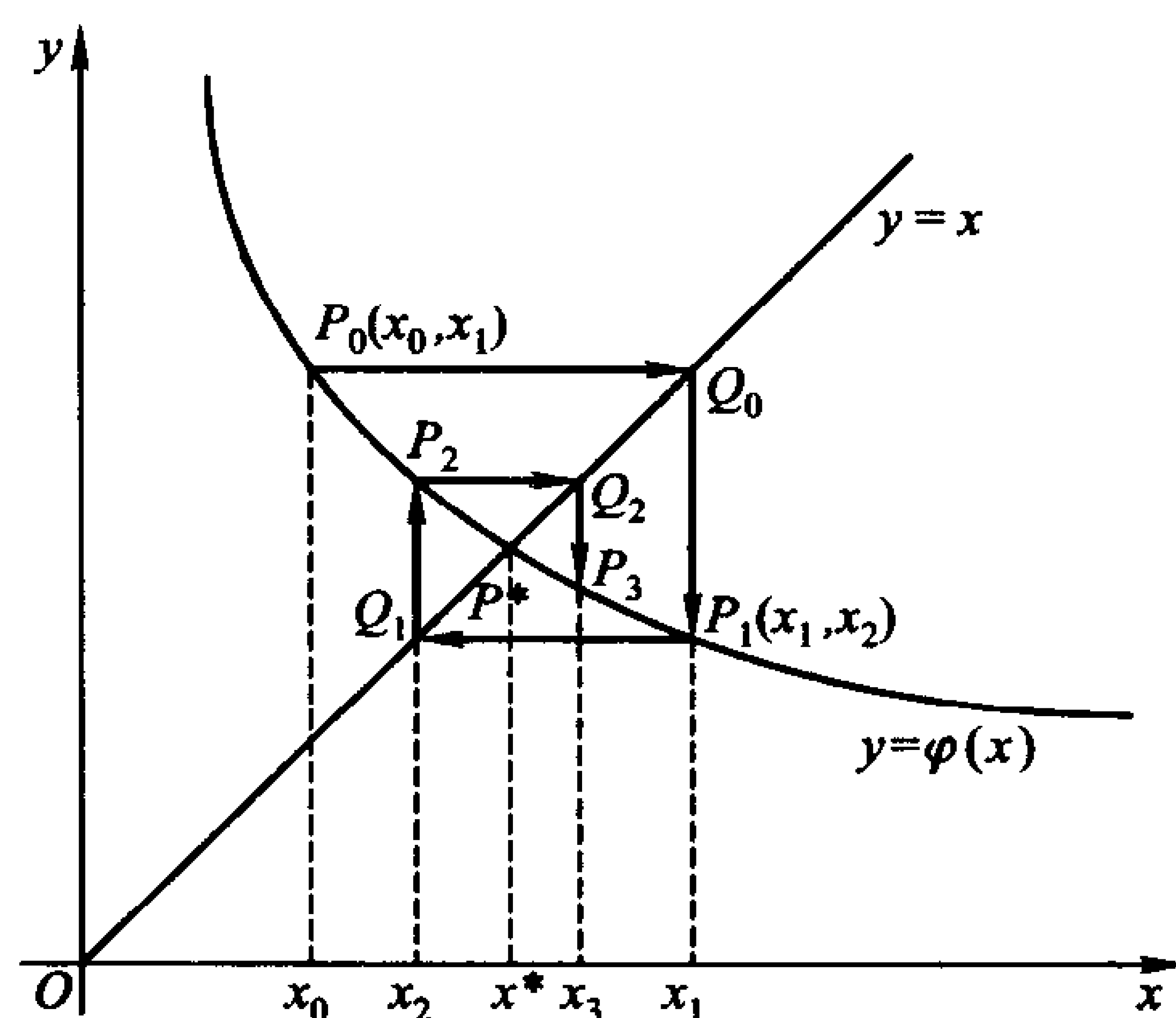


图 2-10 $\{x_n\}$收敛

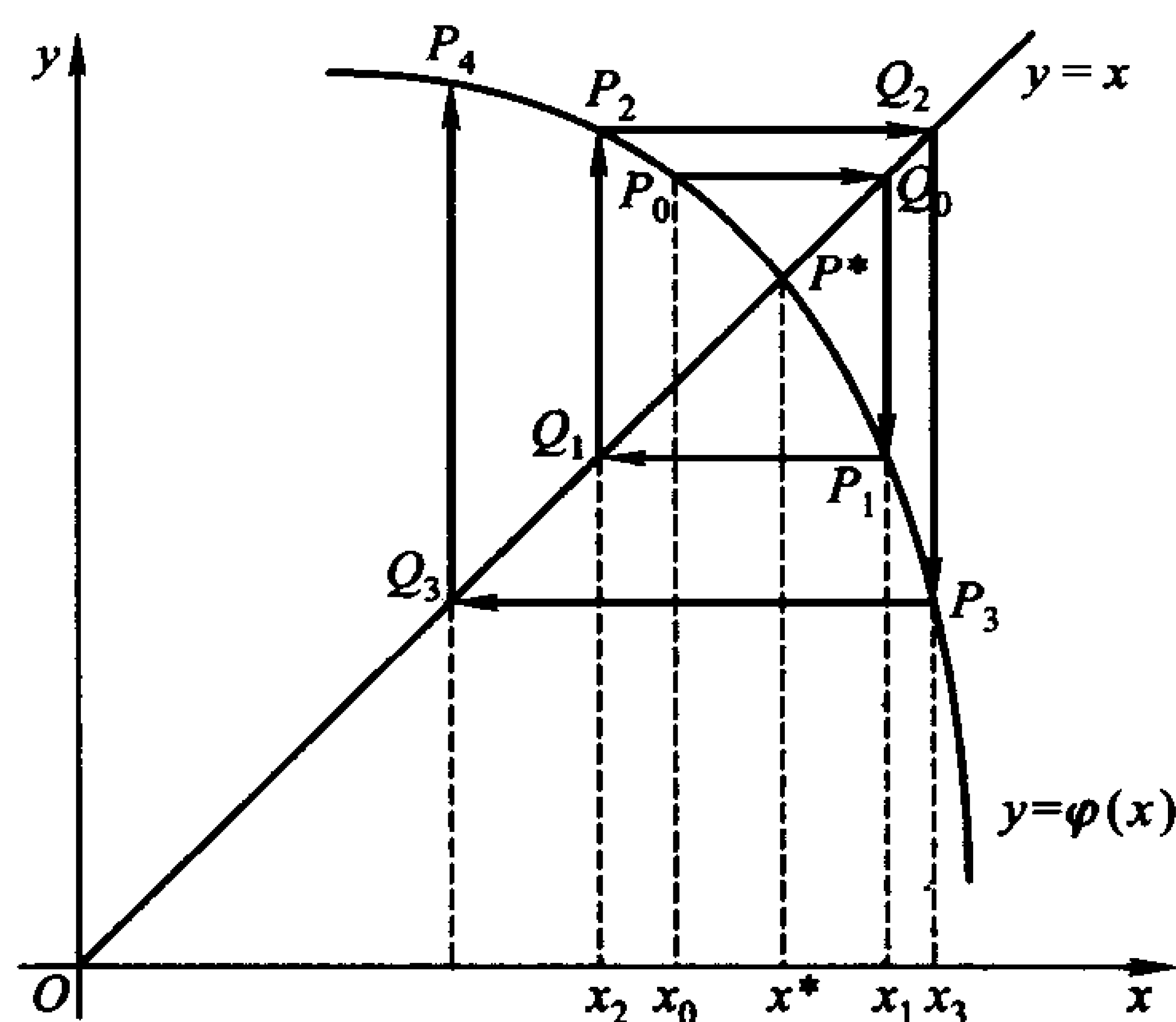


图 2-11 $\{x_n\}$不收敛

2.4.4 迭代法的收敛性

从例 2.4.1 我们可以看出,用同一个方程构成的不同的迭代公式,所得到的迭代序列$\{x_n\}$的敛散性不同,且收敛的速度也不同. 由迭代法的几何解释我们可以看出迭代序列$\{x_n\}$收敛是有条件的. 下面我们来证明一个迭代序列收敛性的定理.

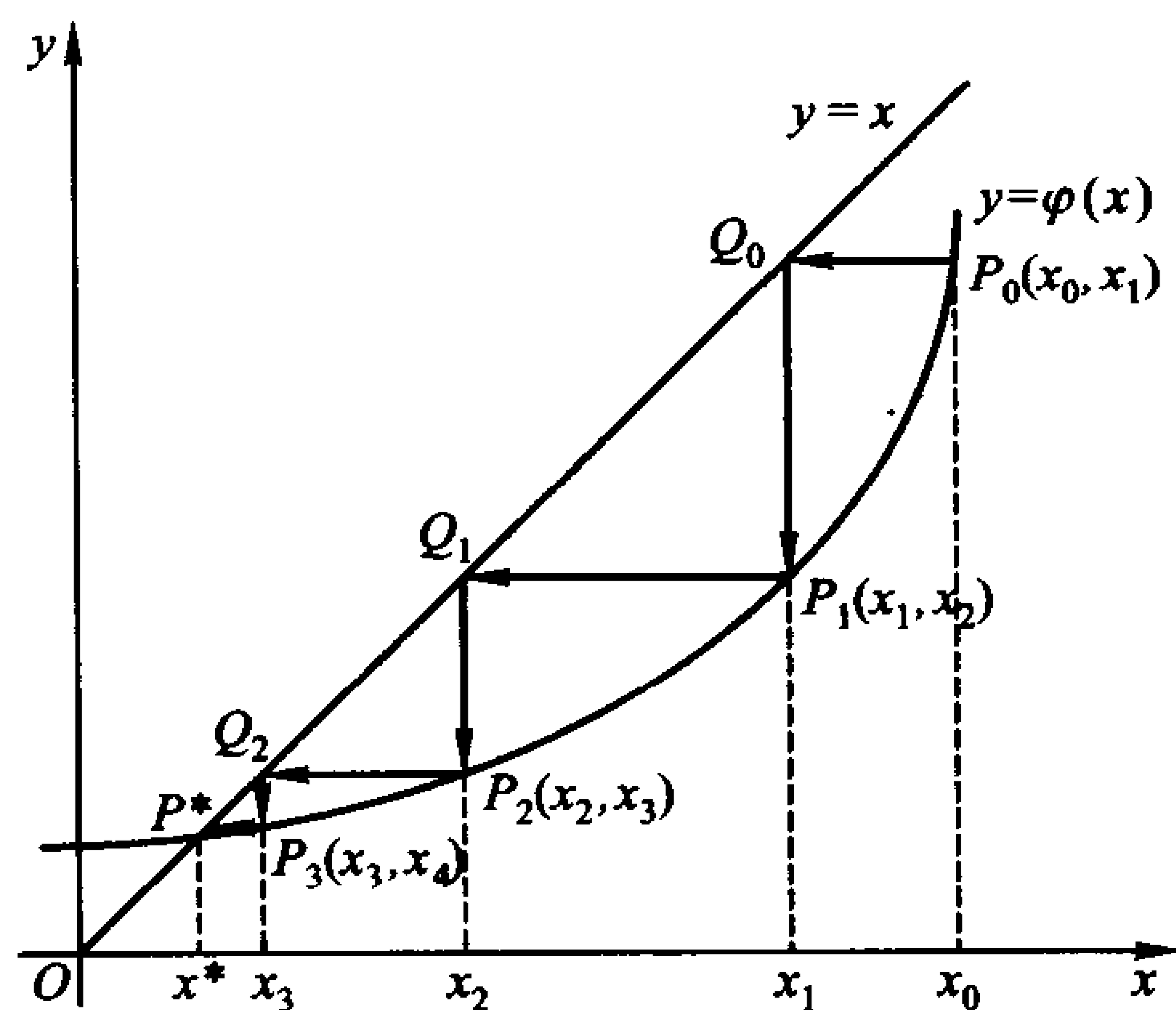


图 2－12 $\{x_n\}$收敛

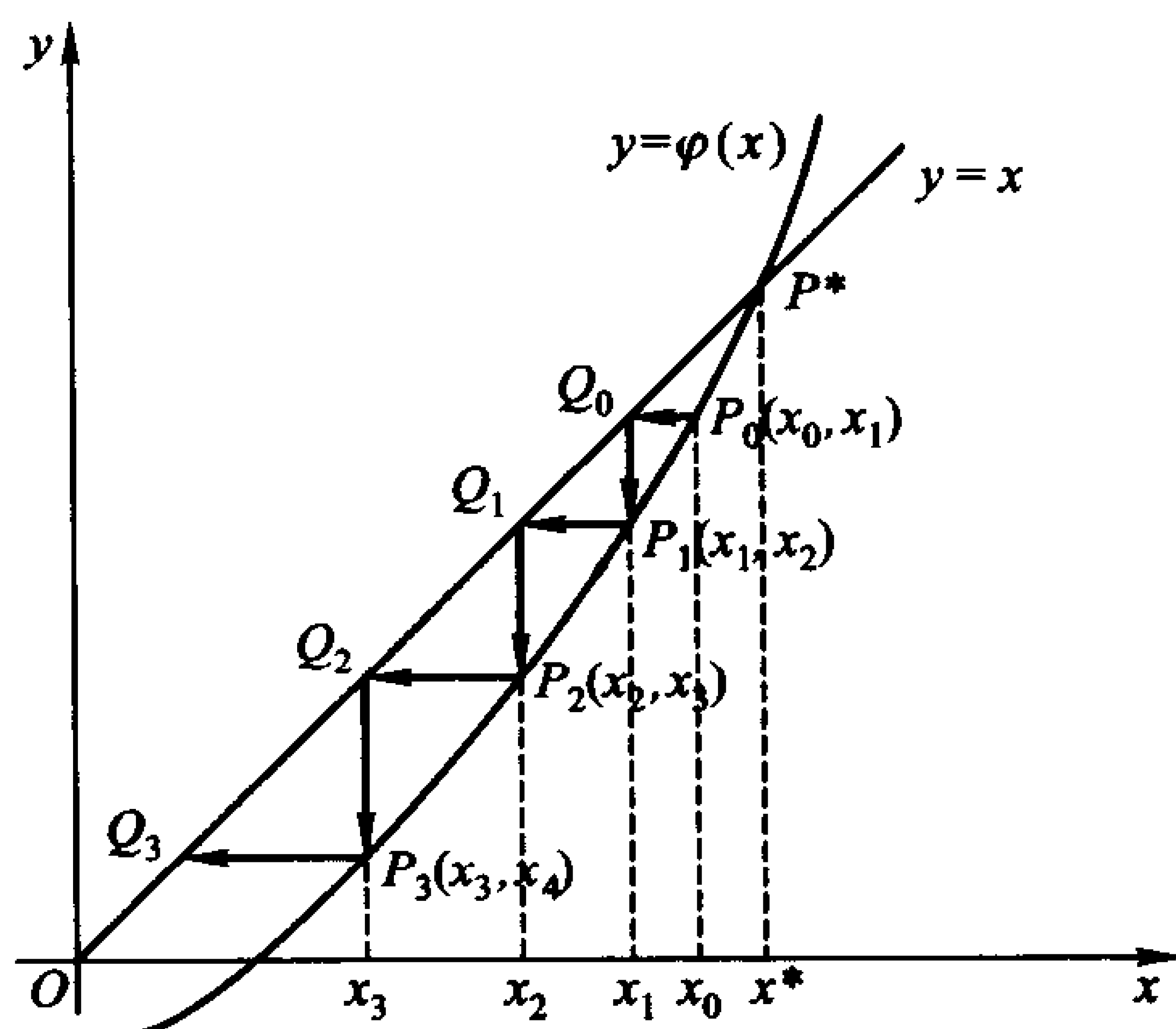


图 2－13 $\{x_n\}$不收敛

定理 2.2 设函数 $y=\varphi(x)$在闭区间$[a,b]$上具有一阶连续导数，且当 $x\in[a,b]$时，$y\in[a,b]$. 若存在正常数 $L<1$，使得对任意 $x\in[a,b]$有 $|\varphi'(x)|\leqslant L$，则

(1) 方程 $x=\varphi(x)$在$[a,b]$上有唯一解 x^*；

(2) 对于任意的初始值 $x_0\in(a,b)$，迭代公式 $x_{k+1}=\varphi(x_k)$ $(k=0,1,2,\cdots)$ 产生的序列$\{x_k\}$收敛于 x^*；

(3) $|x_{k+1}-x^*|\leqslant L|x_k-x^*|\ (k=1,2,\cdots)$; (2.5)

(4) $|x_k-x^*|\leqslant \frac{L^k}{1-L}|x_1-x_0|\ (k=1,2,\cdots)$; (2.6)

(5) $\lim\limits_{k\to\infty}\frac{x_{k+1}-x^*}{x_k-x^*}=\varphi'(x^*)$.

证明 (1)首先证明 $x=\varphi(x)$ 解的存在性和唯一性.

作函数 $h(x)=x-\varphi(x)$,由 $\varphi(x)$ 的连续性知 $h(x)$ 在 $[a,b]$ 上连续,且 $h(a)=a-\varphi(a)\leqslant 0$, $h(b)=b-\varphi(b)\geqslant 0$,故至少存在一个根 $x^*\in(a,b)$,使得 $h(x^*)=0$,即 $x^*=\varphi(x^*)$.

假设存在两个根 $x_1^*,x_2^*\in(a,b)$,使 $x_1^*=\varphi(x_1^*)$, $x_2^*=\varphi(x_2^*)$ 皆成立. 两式相减,利用中值定理,有

$$|x_1^*-x_2^*|=|\varphi(x_1^*)-\varphi(x_2^*)|\leqslant L|x_1^*-x_2^*|.$$

因为 $L<1$,所以 $x_1^*=x_2^*=x^*$.

(2) 其次证明 $\{x_k\}$ 的收敛性.

由迭代公式(2.4)有

$$\begin{aligned}|x^*-x_k|&=|\varphi(x^*)-\varphi(x_{k-1})|=|\varphi'(\xi_{k-1})||x^*-x_{k-1}|\\&\leqslant L|x^*-x_{k-1}|\leqslant\cdots\leqslant L^k|x^*-x_0|.\end{aligned}$$

因为 $L<1$,所以 $\lim\limits_{k\to\infty}x_k=x^*$.

(3) 再证误差公式(2.5)和(2.6).

对于任意的正整数 k,有

$$|x^*-x_{k+1}|=|\varphi(x^*)-\varphi(x_k)|=|\varphi'(\xi_k)||x^*-x_k|\leqslant L|x^*-x_k|,$$

其中 ξ_k 在 x^* 与 x_k 之间. 同理得

$$|x_{k+1}-x_k|\leqslant L|x_k-x_{k-1}|.$$

但是

$$\begin{aligned}|x_{k+1}-x_k|&=|(x^*-x_k)-(x^*-x_{k+1})|\geqslant|x^*-x_k|-|x^*-x_{k+1}|\\&\geqslant(1-L)|x^*-x_k|,\end{aligned}$$

从而有

$$|x^*-x_k|\leqslant\frac{1}{1-L}|x_{k+1}-x_k|\leqslant\frac{L}{1-L}|x_k-x_{k-1}|\ (k=1,2,\cdots). \quad (2.7)$$

继续利用递推式,最后得到 $|x^*-x_k|\leqslant\frac{L^k}{1-L}|x_1-x_0|\ (k=1,2,\cdots)$.

(4) 因为 $x_{k+1}-x^*=\varphi(x_k)-\varphi(x^*)=\varphi'(\xi_1)(x_k-x^*)$(其中 ξ_1 在 x_k 与 x^* 之间),则 $\frac{x_{k+1}-x^*}{x_k-x^*}=\varphi'(\xi_1)$,因此 $\lim\limits_{k\to\infty}\frac{x_{k+1}-x^*}{x_k-x^*}=\varphi'(x^*)$. 证毕.

定理 2.3 设方程 $x=\varphi(x)$ 在闭区间 $[a,b]$ 上有根 x^*,函数 $y=\varphi(x)$ 在 $[a,b]$ 上具有一阶连续导数,且当 $x\in[a,b]$ 时,有 $|\varphi'(x)|\geqslant L$,则对于任意初始值 $x_0\in(a,b)$ 且 $x_0\neq x^*$,迭代公式 $x_{k+1}=\varphi(x_k)$ $(k=0,1,2,\cdots)$ 产生的序列 $\{x_k\}$ 发散.

定义 2.1 设方程 $x=\varphi(x)$ 根 x^* 的某个邻域为 $U(x^*,\delta)=\{x\,|\,|x-x^*|<\delta,\delta>0\}$. 如果对于任意初始值 $x_0\in U(x^*,\delta)$,迭代公式 $x_{k+1}=\varphi(x_k)$ $(k=0,1,2,\cdots)$ 产生的迭代序列 $\{x_k\}$ 收敛,那么称 $\{x_k\}$ 是**局部收敛的**.

值得注意的是对大范围的含根区间 $[a,b]$,定理 2.4.1 的条件不一定成立. 实际上,利用迭代法总是在根 x^* 的邻近进行的. 因为只要 $\varphi(x)$ 在 x^* 的一个邻域内连续且 $|\varphi'(x^*)|<1$,则由连续函数的性质可知,对于该邻域内的任意初始值 x_0,迭代序列 $\{x_k\}$ 就收敛于 x^*.

定理 2.4 设方程 $x=\varphi(x)$ 有根 x^*,且在某个邻域 $U(x^*,\delta)=\{x\,|\,|x-x^*|<\delta,\delta>0\}$ 内函数 $y=\varphi(x)$ 在闭区间 $[a,b]$ 上具有一阶连续导数,迭代公式 $x_{k+1}=\varphi(x_k)$ $(k=0,1,2,\cdots)$ 产生迭代序列 $\{x_k\}$,则 (1) $|\varphi'(x^*)|<1$ 时,迭代序列 $\{x_k\}$ 局部收敛;(2) $|\varphi'(x^*)|>1$ 时,迭代序列 $\{x_k\}$ 发散.

实际计算中,误差公式(2.6)不仅可以用来估计迭代 k 次时的误差,还可以用来估计达到给定的精度 ε 要求所需迭代次数 k. 如果欲使 $|x^*-x_k|\leqslant\varepsilon$,只要

$$\frac{L^k}{1-L}|x_1-x_0|\leqslant\varepsilon,$$

从而迭代次数 k 满足

$$k\geqslant\left|\frac{\ln\varepsilon+\ln(1-L)-\ln|x_1-x_0|}{\ln L}\right|. \tag{2.8}$$

例 2.4.2 证明对任何初始值 $x_0\in\mathbf{R}$,由迭代公式 $x_{k+1}=\cos x_k, k=0,1,2,\cdots$ 所产生的迭代序列 $\{x_k\}$ 都收敛于方程 $x=\cos x$.

证 设 $\varphi(x)=\cos x$,则 $\varphi'(x)=-\sin x$ 在 $(-\infty,+\infty)$ 内连续.

(1) 先考虑区间 $[-1,1]$. 当 $x\in[-1,1]$ 时,$\varphi(x)=\cos x\in[-1,1]$,且 $|\varphi'(x)|=|-\sin x|<1$,根据定理 2.4.1 知,对于任何初始值 $x_0\in[-1,1]$,迭代公式 $x_{k+1}=\cos x_k, k=0,1,2,\cdots$ 产生的序列 $\{x_k\}$ 都收敛于方程 $x=\cos x$ 的根 x^*.

(2) 对于任何初始值 $x_0\in\mathbf{R}$,有 $x_1=\cos x_0\in[-1,1]$. 将 x_1 看成新的迭代初始值,则由(1)可知,由迭代公式 $x_{k+1}=\cos x_k, k=0,1,2,\cdots$ 产生的序列 $\{x_k\}$ 都收敛于方程 $x=\cos x$ 的根 x^*.

例 2.4.3 利用适当的迭代公式证明

$$\lim_{k\to\infty}\underbrace{\sqrt{a+\sqrt{a+\sqrt{a+\cdots+\sqrt{a}}}}}_{k\text{个}a}=\frac{1+\sqrt{1+4a}}{2}\ (\text{其中 } a=2,3,4,5,\cdots,n).$$

证 考虑迭代公式 $\begin{cases}x_0=0,\\ x_{k+1}=\sqrt{a+x_k},k=0,1,2,\cdots.\end{cases}$

设 $\varphi(x)=\sqrt{a+x}$，当 $x\in[0,a]$时，则 $|\varphi'(x)|=\dfrac{1}{2\sqrt{a+x}}<1$ $(a=2,3,4,5,\cdots,n)$，且 $\varphi(x)\in[\varphi(0),\varphi(a)]=[\sqrt{a},\sqrt{2a}]\subset[0,a]$，根据定理 2.4.1 知，对于初始值 $x_0=0$，迭代公式 $x_{k+1}=\sqrt{a+x_k},k=0,1,2,\cdots$产生的序列$\{x_k\}$收敛于方程 $x=\sqrt{a+x}$的根 $x^*=\dfrac{1+\sqrt{1+4a}}{2}$，即$\lim\limits_{k\to\infty}x_k=\dfrac{(1+\sqrt{1+4a})}{2}$.

实际要在区间 $a\leqslant x\leqslant b$ 内确定 L，使$|\varphi'(x)|\leqslant L$ 是不太方便的，并且 L 不易求得. 但注意到估计式(2.7)，当 $0<L<1$ 越小时，迭代序列$\{x_k\}$收敛越快. 只要相邻两次迭代的偏差$\{x_k-x_{k-1}\}$足够小时，就可以保证近似解 x_k 有足够的精度. 因此，上机计算时，常采用条件$|x_k-x_{k-1}|\leqslant\varepsilon$ 来控制停机；当$|x_k|$的数量级较大时，也可以用相对误差$\dfrac{|x_k-x_{k-1}|}{|x_k|}\leqslant\varepsilon$ 作为迭代终止的条件；为避免出现无限制的迭代，也可以规定一个最大的迭代次数(见迭代法的 MATLAB 程序 2).

例 2.4.4 为求方程$f(x)=x^3-x^2-1=0$ 在 $x_0=1.5$ 附近的近似根，可将方程改写成下列等价形式，并建立相应的迭代公式

(1) 取 $x=1+\dfrac{1}{x^2}$，迭代公式为 $x_{k+1}=1+\dfrac{1}{x_k^2}$；

(2) 取 $x=\sqrt[3]{1+x^2}$，迭代公式为 $x_{k+1}=\sqrt[3]{1+x_k^2}$；

(3) 取 $x=\dfrac{1}{\sqrt{x-1}}$，迭代公式为 $x_{k+1}=\dfrac{1}{\sqrt{x_k-1}}$；

(4) 取 $x=\sqrt{x^3-1}$，迭代公式为 $x_{k+1}=\sqrt{x_k^3-1}$.

试分析每一种迭代公式的收敛性，并选择收敛的迭代公式求方程的根，精确到 4 位有效数字，并比较它们收敛的速度.

解 (1)输入程序

```
>>x=solve('x^3-x^2-1=0')
```

运行后输出精确解

```
x1 =1/6*(116+12*93^(1/2))^(1/3)+2/3/(116+12*93^(1/2))^
(1/3)+1/3,
x2 = -1/12*(116+12*93^(1/2))^(1/3) -1/3/(116+12*93^
```

```
(1/2))^(1/3)+1/3+1/2*i*3^(1/2)*(1/6*(116+12*93^(1/2))^
(1/3)-2/3/(116+12*93^(1/2))^(1/3)),
    x3 = -1/12*(116+12*93^(1/2))^(1/3)-1/3/(116+12*93^(1/2))^
(1/3)+1/3-1/2*i*3^(1/2)*(1/6*(116+12*93^(1/2))^(1/3)-2/3/
(116+12*93^(1/2))^(1/3))
```

(2) 编写用迭代公式 $x_{k+1}=\varphi(x_k)$计算方程的近似根的 MATLAB 主程序

```
function diedai9
n=1000;x0=1.5; x=zeros(n,1); jwx=zeros(n,1); xwx=zeros(n,1);
for k=1:n
  x1=fun(x0); x(k)=x1; jwx(k)=abs(x1-x0); xwx(k)= jwx(k)/x(k);
  if jwx(k)<1.e-4,return
    else
    k=k+1;x0=x1;
  end
end
m=1:k;x=x(1:k,1); jwx=jwx(1:k,1); xwx=xwx(1:k,1); [m',x, jwx, xwx]
```

(3) 取 $x=1+\frac{1}{x^2}$,即 $\varphi(x)=1+\frac{1}{x^2}$,则 $\varphi'(x)=-\frac{2}{x^3}\neq 0$. 输入程序

```
>> x=1.4:0.01:1.5;dfai= abs(-2./(x.^3));df=max(dfai)
```

运行后输出结果

```
df =
   0.72886297376093
```

则 $x\in[1.4,1.5]$,$|\varphi'(x)|<1$,故迭代公式 $x_{k+1}=1+\frac{1}{x_k^2}$在$[1.4,1.5]$内收敛.

建立 M 文件

```
function y=fun(x)
     y=1+1/(x^2);
```

输入文件名

```
>> diedai9
```

运行后将屏幕显示的结果列入表 2-5:

表 2-5

k	x_k	$\|x_{k+1}-x_k\|$	$\frac{\|x_{k+1}-x_k\|}{\|x_{k+1}\|}$
1	1.444 4	0.055 6	0.038 5
2	1.479 3	0.034 8	0.023 6
3	1.457 0	0.022 3	0.015 3

续表

k	x_k	$\lvert x_{k+1}-x_k\rvert$	$\frac{\lvert x_{k+1}-x_k\rvert}{\lvert x_{k+1}\rvert}$
4	1.471 1	0.014 1	0.009 6
5	1.462 1	0.009 0	0.006 1
6	1.467 8	0.005 7	0.003 9
7	1.464 2	0.003 6	0.002 5
8	1.466 5	0.002 3	0.001 6
9	1.465 0	0.001 5	0.001 0
10	1.465 9	0.000 9	0.000 6
11	1.465 3	0.000 6	0.000 4
12	1.465 7	0.000 4	0.000 3

取 $x_0=1.5$,用迭代法 $x_{k+1}=1+\frac{1}{x_k^2}$ $(k=0,1,2,\cdots)$计算,满足精确到 4 位有效数字的近似根 1.466.

(4) 取 $x=\sqrt[3]{1+x^2}$,即 $\varphi(x)=\sqrt[3]{1+x^2}$,则 $\lvert\varphi'(x)\rvert=\left|\frac{2x}{3\cdot\sqrt[3]{(1+x^2)^2}}\right|$. 输入程序

```
>> x =1.4:0.01:1.5;
 dfai = abs (2 * x./(3 * ((1 +x.^2).^2).^(1/3)));df =max(dfai)
```

运行后屏幕显示结果

```
df =0.45576862590883
```

所以,$\lvert\varphi'(x)\rvert<1$,$x\in[1.4,1.5]$,故迭代公式 $x_{k+1}=\sqrt[3]{1+x_k^2}$在$[1.4,1.5]$内收敛.

建立 M 文件

```
function y = fun(x)
         y =(1 +x^2)^(1/3);
```

输入文件名

```
>> diedai9
```

运行后将屏幕显示的结果列入表 2-6:

表 2-6

k	x_k	$\|x_{k+1}-x_k\|$	$\frac{\|x_{k+1}-x_k\|}{\|x_{k+1}\|}$
1	1.481 2	0.018 8	0.012 7
2	1.472 7	0.008 5	0.005 8
3	1.468 8	0.003 9	0.002 6
4	1.467 0	0.001 8	0.001 2
5	1.466 2	0.000 8	0.000 5
6	1.465 9	0.000 4	0.000 2

取 $x_0=1.5$,用迭代法 $x_{k+1}=\sqrt[3]{1+x_k^2}$ $(k=0,1,2,\cdots)$计算,满足精确到4位有效数字的近似根1.466.

(5) 取 $x^2=\frac{1}{x-1}$,即 $\varphi(x)=\frac{1}{\sqrt{x-1}}$,则 $\varphi'(x)=-\frac{1}{2\sqrt{(x-1)^3}}\neq 0$,输入程序

```
>> x =1.4:0.01:1.5;
 dfai = abs ( -1./(2*((x-1).^3).^(1/2)));df =max(dfai),
 x1 =1.46557123187677;
 dfai1 = abs ( -1./(2*((x1 -1).^3).^(1/2)))
```

运行后屏幕显示结果

```
df =1.97642353760524,dfai1 =1.57394951785238
```

即 $|\varphi'(x)|>1, x\in[1.4,1.5]$,故迭代公式 $x_{k+1}=\frac{1}{\sqrt{x_k-1}}$ 在[1.4,1.5]是发散的.

(6) 取 $x=\sqrt{x^3-1}$,即 $\varphi(x)=\sqrt{x^3-1}$,则 $\varphi'(x)=\frac{3x^2}{2\cdot\sqrt{x^3-1}}\neq 0$,输入程序

```
>>x =1.4:0.01:1.5;
 dfai = abs ((3*x.^2)./(2*(x.^3 -1).^(1/2)));df =max(dfai),
 x1 =1.46557123187677;
 dfai1 = abs ((3*x1.^2)./(2*(x1.^3 -1).^(1/2)))
```

运行后屏幕显示结果

```
df =2.22625080809653, dfai1 =2.19835684781515
```

由此可见,迭代公式 $x_{k+1}=\sqrt{x_k^3-1}$ 在[1.4,1.5]是发散的. 迭代公式 $x_{k+1}=\sqrt[3]{1+x_k^2}$ 比 $x_{k+1}=1+\frac{1}{x_k^2}$ 收敛的速度快.

2.4.5 迭代法的 MATLAB 程序 2

(一) 迭代法的 MATLAB 程序 2

> **迭代法的 MATLAB 程序 2**
> 已知初始值 x_0、精度 *tol*、最大迭代次数 *ddmax* 和迭代公式 $x_{k+1}=\phi(x_k)$，求满足精度 *tol* 的近似根 x_k 和迭代次数 k.

根据迭代公式(2.4)和已知条件，编写一个名为 diedai2.m 的 M 文件.

```
function [k,piancha,xdpiancha,xk,yk] = diedai2(x0,tol,ddmax)
x(1) = x0;
for i = 1: ddmax
    x(i +1) = fun(x(i));piancha = abs(x(i +1) - x(i));
    xdpiancha = piancha/( abs(x(i +1)) + eps);i = i +1;
    xk = x(i);yk = fun(x(i)); [(i -1) piancha xdpiancha xk yk]
 if (piancha < tol) |(xdpiancha < tol)
          k = i -1; xk = x(i);
    return;
end
end
  if i > ddmax
        disp('迭代次数超过给定的最大值 ddmax')
        k = i -1;xk = x(i);yk = fun(x(i)); [(i -1) piancha xdpiancha
        xk yk]
        return;
end
P = [(i -1),piancha,xdpiancha,xk,yk]';
```

(二) 使用迭代法的 MATLAB 程序 2 求方程 $f(x)=0$ 的近似根(精度为 *tol*)的步骤

步骤 1 把给定的方程 $f(x)=0$ 改写成容易迭代的形式 $x=\varphi(x)$.

步骤 2 建立 M 文件 fun.m

```
function y = fun(x)
```

$$y=\varphi(x);$$

步骤 3 保存名为 diedai2.m 的 M 文件.

步骤 4 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk,yk] = diedai2(x0,tol,ddmax)
```

其中输入的量:初始值 x_0，允许最大迭代次数 *ddmax* (迭代次数达到时运行后输出信息'迭代次数超过给定的最大值 ddmax')及(相对)精度 *tol* 都是具体给定的数值，然后运行即可运行后输出迭代次数 k、满足精度 *tol* 的根的近似根 x_k、它的函数值 $y_k=\varphi(x_k)$、相邻两次迭代的偏差 $piancha=|x_k-x_{k-1}|$ 和它的相对误差 $xdpiancha=|x_k-x_{k-1}|/|x_k|$ 的值(参见例 2.4.5).

例 2.4.5 求 $x^5-3x+1=0$ 在 0.3 附近的根，精确到 4 位小数.

解 (1) 构造迭代公式 $x_k=\varphi(x_k)$.

改写原方程为等价方程 $x=(x^5+1)/3$. 这时 $\varphi(x)=(x^5+1)/3$, 初始值为 $x_0=0.5$, 迭代公式

$$x_{k+1}=(x_k^5+1)/3 \quad (k=0,1,2,\cdots).$$

(2) 利用迭代法的 MATLAB 程序 2 计算精确到 4 位小数的根的近似值 y 和迭代次数 k.

① 保存名为 diedai2.m 的 M 文件.

② 建立 M 文件 fun.m

```
function y = fun(x)
        y = (x^5 +1)/3;
```

③ 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk,yk] = diedai2(0.3,1e-4,100)
```

④ 运行后输出的结果

```
[k,piancha,xdpiancha,xk,yk] =
      0   0.03414   0.10218   0.30000   0.33414
      1   0.03414   0.10218   0.33414   0.33472
      2   0.00058   0.00173   0.33472   0.33473
      3   0.00001   0.00004   0.33473   0.33473
k =3;piancha =1.206089525390697e-005;
xdpiancha =3.603129477781680e-005;
xk =0.3347; yk =0.3347.
```

注意 (1) 当迭代次数超过给定的最大值 *ddmax* = 1 时,运行程序[k,piancha,xdpiancha,xk,yk] = diedai2(0.3,1e-4,1),得根的近似值 $y=0.334\ 72$ 不满足精度 1e-4.

(2) 若将初始值改写 30 后输入如下程序

```
>> [k,piancha,xdpiancha,xk,yk] = diedai2(30,1e-4,100)
```

则运行后输出的结果

```
[k ,piancha,xdpiancha,xk,yk] =
   1  8.1000e+006  1.0000e+000  8.1000e+006  1.1623e+034
   2  1.1623e+034  1.0000e+000  1.1623e+034  7.0697e+169
   3  7.0697e+169  1.0000e+000  7.0697e+169        Inf
   4          Inf          NaN          Inf         Inf
   5          NaN          NaN          Inf         Inf
  ................................................
  99          NaN          NaN          Inf         Inf
 100          NaN          NaN          Inf         Inf
迭代次数超过给定的最大值 ddmax
```

```
Warning: One or more output arguments not assigned during call to 'diedai2'.
piancha =NaN;  xdpiancha =NaN;  xk =Inf;  yk =Inf.
```

其中 NaN 表示**不定值**，Inf 表示**正无穷大**.此迭代数列发散.这说明**迭代数列的敛散性不但与迭代公式有关,还与初始值有关**.

例 2.4.6 求 $5x-e^x=0$ 的一个实根,精确到 4 位小数.

解 (1) 先确定方程 $5x-e^x=0$ 的实根的分布情况.

适当调整 x 的范围,在 MATLAB 工作窗口输入程序

```
>> e=2.71828;x=0:0.03:3;y=5.*x-e.^x;
plot(x,y),
grid,gtext('y=5x-exp(x)')
```

画出函数 $f(x)=5x-e^x$ 的图像如图 2-14. 从曲线 $f(x)$ 与 x 轴的交点可以看出,在图形所表示的 x 的范围内,此方程有 2 个实根,分别位于 $x=0.25,2.5$ 附近.

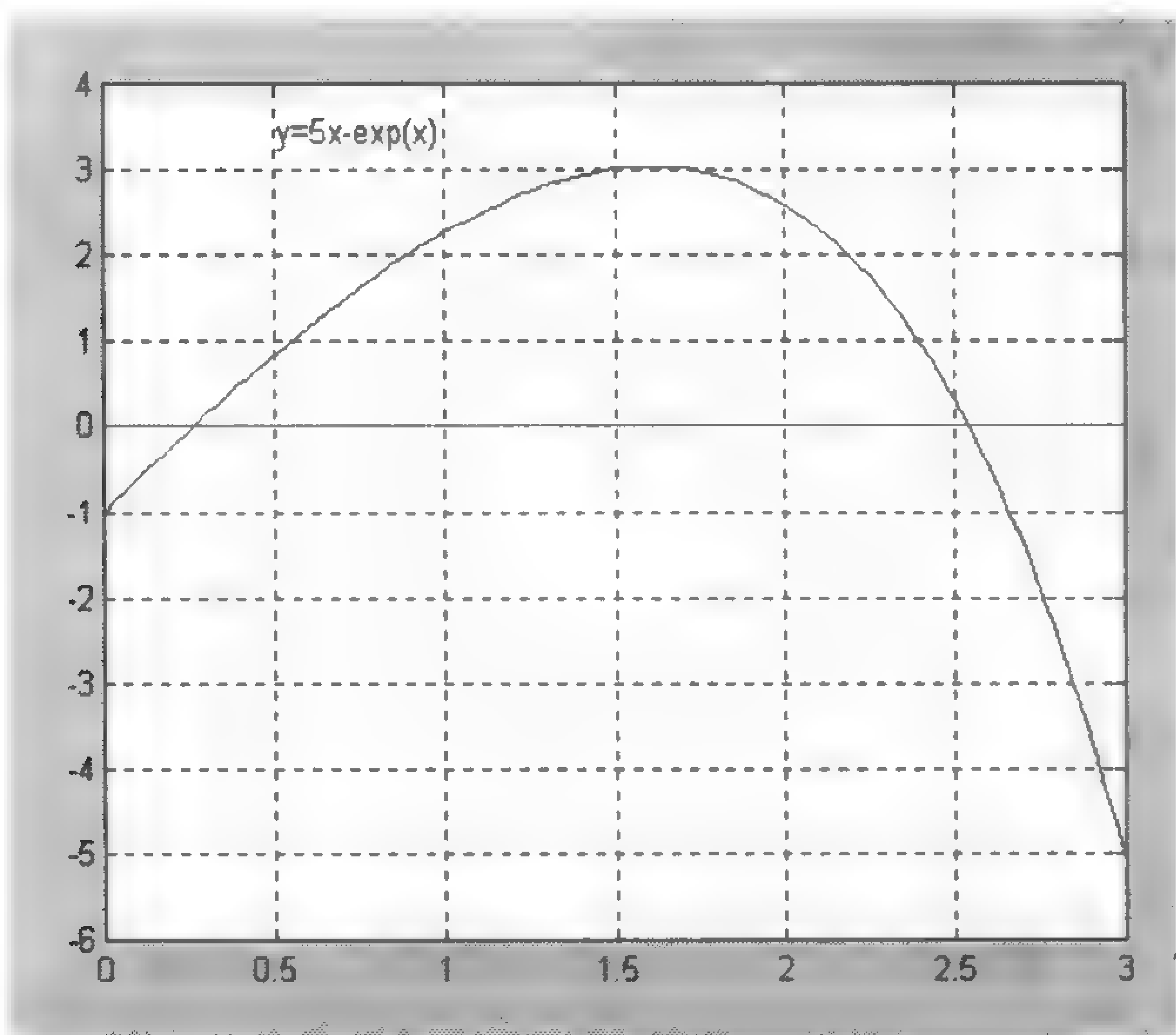


图 2-14

(2) 构造迭代公式 $x_k=\varphi(x_k)$.

改写原方程为等价方程 $x=e^x/5$,这时 $\varphi(x)=e^x/5$,取初始值为 $x_0=0.25$,迭代公式

$$x_k=e^{x_k}/5 \quad (k=0,1,2,\cdots).$$

(3) 利用迭代法的 MATLAB 程序 2 计算精确到 4 位小数的根的近似值 y 和迭代次数 k.

① 保存名为 diedai2.m 的 M 文件.

② 建立 M 文件 fun.m

```
function y = fun(x)
            e = exp(1); y = e^x/5;
```

③ 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk,yk] = diedai2(0.25,1e-4,100)
```

④ 运行后输出的结果(略). 运行次数为 $k=5$,精度为 1e-4 的根的近似值为 $x_k=0.259\,2$;其函数值为 $y_k=0.259\,2$;偏差为 *piancha* =3.046 6e-005;其绝对误差为 *xdpiancha* = 1.175 6e-004.

例 2.4.7 求 $2e^{-x}-x=0$ 在 0.85 附近的一个近似根,要求精度 $\varepsilon=10^{-6}$.

解 (1)求迭代公式.

将方程 $2e^{-x}-x=0$ 化为同解方程 $x=2e^{-x}$,得到迭代公式

$$x_{k+1}=2e^{-x_k}\ (k=1,2,3,\cdots).$$

(2) 用搜索法判断根所在的区间.

在 MATLAB 工作窗口输入程序

```
>> x = 0.85:0.01:0.90, e = exp(1); f = 2.*e.^(-x)-x
```

运行后输出结果

```
x = 0.8500    0.8600    0.8700    0.8800    0.8900    0.9000
f = 0.0048   -0.0137   -0.0321   -0.0504   -0.0687   -0.0869
```

可见,所求的根在开区间(0.85,0.86)内.

(3) 建立 M 文件 fun.m

```
function y = fun(x)
         e = 2.718281828; y = 2*e^(-x);
```

(4) 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk,yk] = diedai2(0.85,1e-6,100)
```

(5) 运行后输出的结果经过整理列入表 2-7:

表 2-7 $2e^{-x}-x=0$ 在 0.85 附近的一个近似根

k	*piancha*	*xdpiancha*	x_k	y_k
0	0.004 829 9	0.005 650 1	0.850 000 0	0.854 829 9
1	0.004 829 9	0.005 650 1	0.854 829 9	0.850 711 1
2	0.004 118 8	0.004 841 5	0.850 711 1	0.854 222 2
3	0.003 511 1	0.004 110 3	0.854 222 2	0.851 228 2
⋮	⋮	⋮	⋮	⋮

续表

k	*piancha*	*xdpiancha*	x_k	y_k
51	0.000 001 7	0.000 002 0	0.852 606 3	0.852 604 8
52	0.000 001 4	0.000 001 7	0.852 604 8	0.852 606 1
53	0.000 001 2	0.000 001 4	0.852 606 1	0.852 605 0
54	0.000 001 0	0.000 001 2	0.852 605 0	0.852 605 9
55	0.000 000 9	0.000 001 0	0.852 605 9	0.852 605 2

由表 2 - 7，可见迭代序列收敛. 又因为迭代 $k=55$ 次的函数值 $y_k=0.852\ 605$，且最后相邻两次迭代值的偏差 *piancha* = 8.797 880e - 007；其相对误差为 *xdpiancha* = 1.031 881e - 006. 所以，迭代值 $x_k=0.852\ 606$ 是要求的根的近似值.

例 2.4.8 当 a 取适当值时，圆 $(x-9)^2+y^2=a^2$ 与抛物线 $y=x^2$ 相切，试用迭代法求切点横坐标的近似值，要求不少于 5 位有效数字.

分析 因为两条曲线相切，在切点处的函数值相等且导数相等. 所以可以列出切点横坐标应该满足的关系式，然后用迭代法求之.

解 (1) 求两条曲线相切的条件.

因为 $y=x^2$ 的导数 $y'=2x$；$(x-9)^2+y^2=a^2$ 的导数 y' 满足 $2(x-9)+2y'y=0$，由两条曲线相切的条件，可得

$$2(x-9)+2\cdot 2x\cdot x^2=0，即\ 2x^3+x-9=0.$$

(2) 下面介绍两种找方程的有根区间的方法.

方法 1 作图法.

首先取 x 得较大区间 $[-20,20]$，在工作窗口输入程序

```
>> x = -20:0.1:20; f =2.*x.^3 +x -9; plot(x,f)
grid,gtext('f =2x^3 +x -9')
```

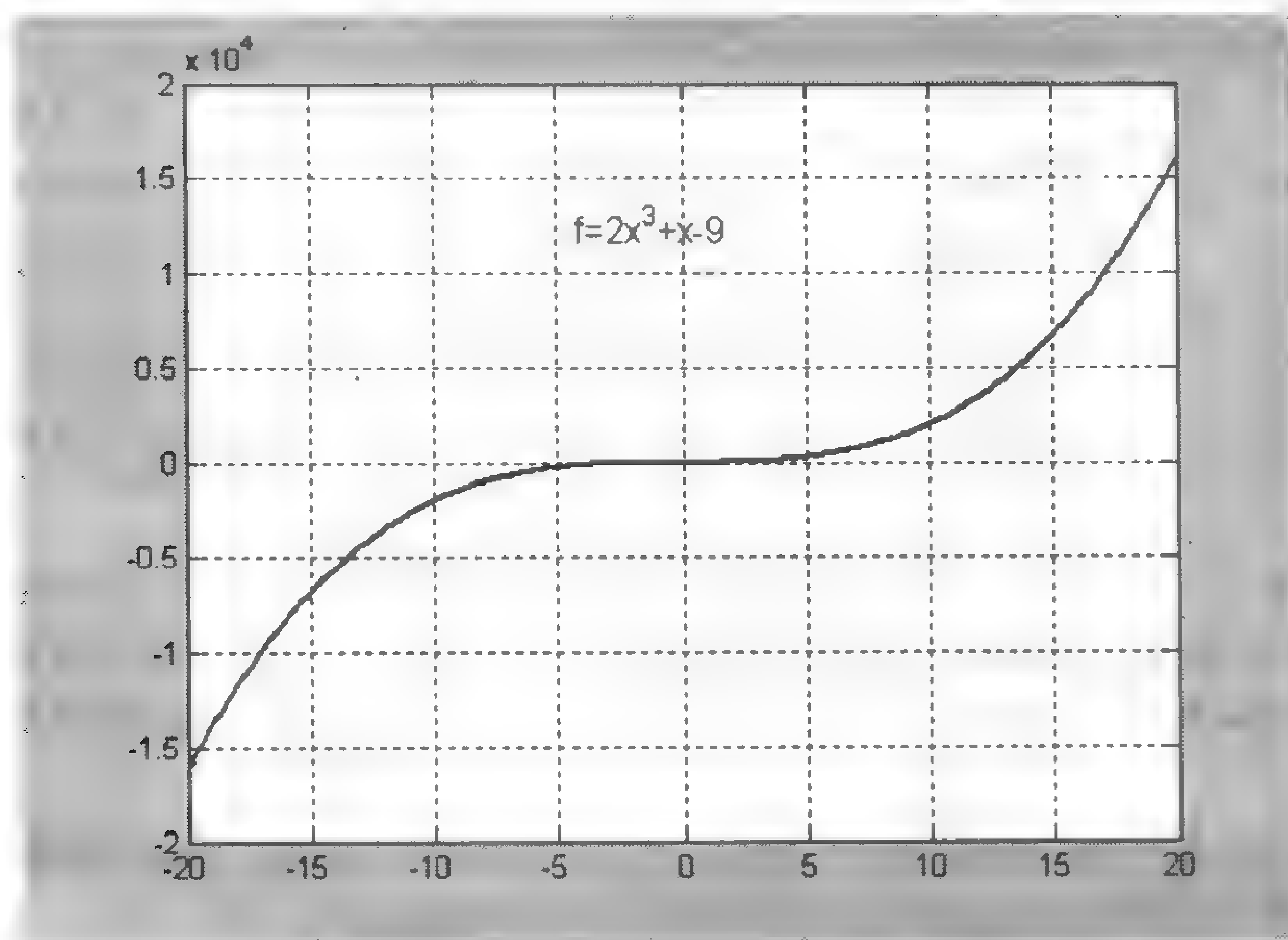
运行后得函数 $f(x)=2x^3+x-9$ 的图形(见图 2 - 15(a)). 由图可见，方程在 $[-5,5]$ 上有根，画出在 $[-5,5]$ 上 $f(x)$ 的图形(见图 2 - 15(b))，可见 $f(x)$ 在 $(1,2)$ 内有唯一实根.

方法 2 试值法.

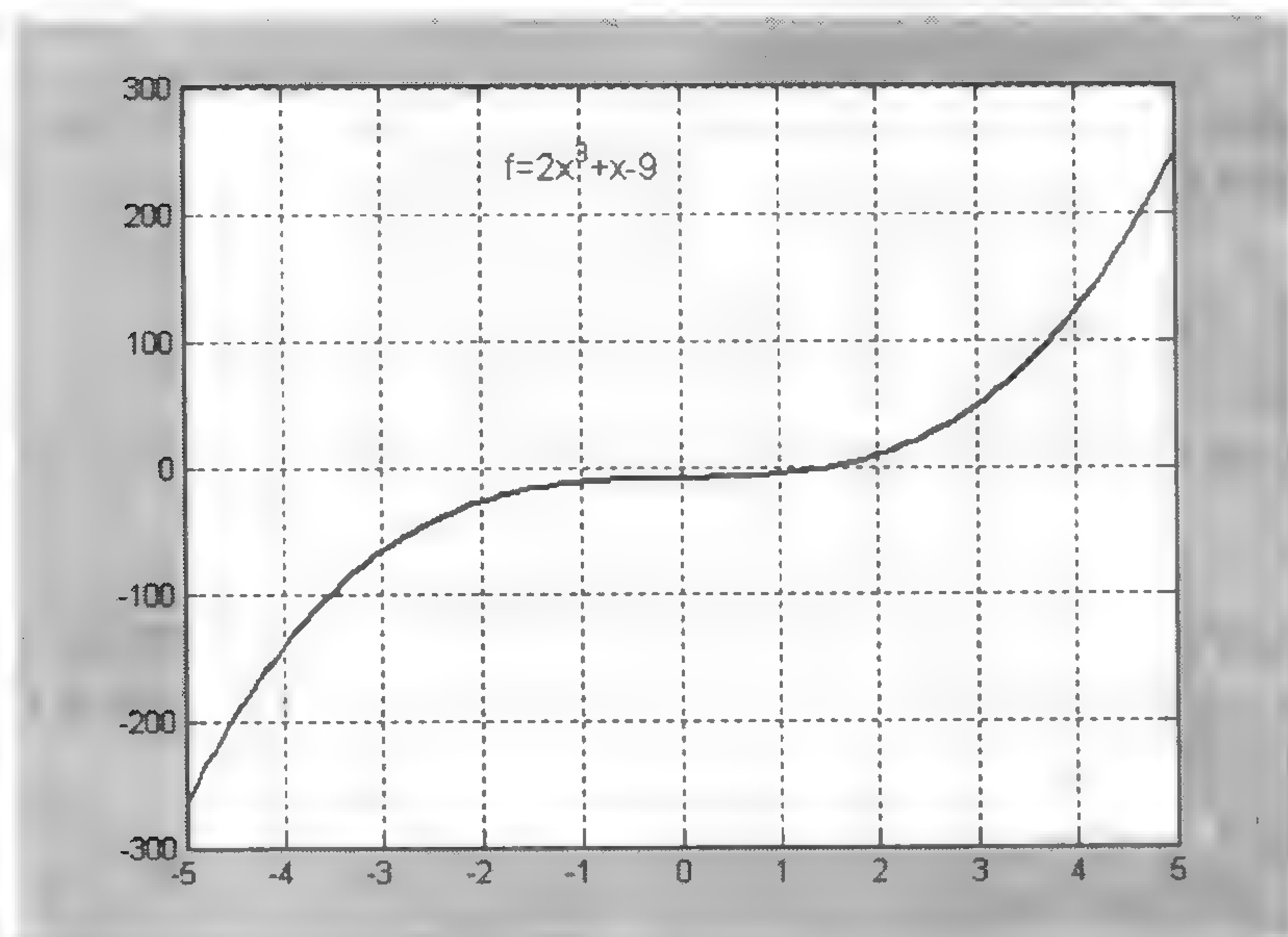
在工作窗口输入程序

```
>> xk = -20:20; fk =2.*xk.^3 +xk -9;[xk;fk]
```

运行后整理得结果填入表 2 - 8. 由此可见，$f(1)=-6<0$，$f(2)=9>0$，因此 $f(x)=0$ 在 $(1,2)$ 内至少有一个实根.


x 10^4
f=2x^3+x-9
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-20
-15
-10
-5
0
5
10
15
20

(a)


f=2x^3+x-9
300
200
100
0
-100
-200
-300
-5
-4
-3
-2
-1
0
1
2
3
4
5

(b)

图 2-15

表 2－8

x_k	－20	－19	－18	－17	－16	－15	－14	－13	－12
函数值 f_k	－16 029	－13 746	－11 691	－9 852	－8 217	－6 774	－5 511	－4 416	－3 477
x_k	－11	－10	－9	－8	－7	－6	－5	－4	－3
函数值 f_k	－2 682	－2 019	－1 476	－1 041	－702	－447	－264	－141	－66
x_k	－2	－1	0	1	2	3	4	5	6
函数值 f_k	－27	－12	－9	－6	9	48	123	246	429
x_k	7	8	9	10	11	12	13	14	15
函数值 f_k	684	1 023	1 458	2 001	2 664	3 459	4 398	5 493	6 756
x_k	16	17	18	19	20				
函数值 f_k	8 199	9 834	11 673	13 728	16 011				

(3) 用迭代法求根.

① 构造迭代公式为 $x_{k+1}=[(9-x_k)/2]^{1/3}, k=0,1,2,\cdots$.

② 取 $x_0=\frac{1+2}{2}=1.5$.

③ 建立并保存下面的 M 文件 fun.m

```
function y = fun(x)
         y = ((9 - x)/2)^(1/3);
```

④ 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk,yk] = diedai2(1.5,1e-4,100)
```

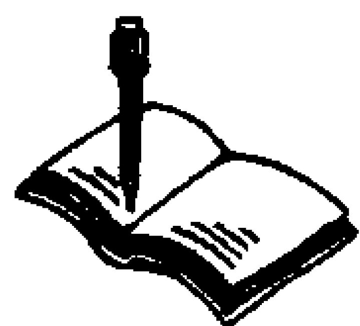
⑤ 运行后整理得结果如下表 2－9：

表 2－9

迭代次数 k	$piancha=\|x_k-x_{k-1}\|$	根的近似值 x_k
1	0.053 62	1.553 62
2	0.003 71	1.549 91
3	0.000 26	1.550 16
4	0.000 02	1.550 14

由于 $piancha=|x_4-x_3|<\frac{1}{2}\times10^{-4}$，故取 $x^*\approx x_4=1.550\ 1$，即可保证有 5 位有

效数字. 因此,两曲线切点的横坐标为 1.550 1.



习 题 2.4

1. 证明对任何初始值 $x_0 \in \mathbf{R}$,由迭代公式

$$x_{k+1} = \frac{x_k}{2} + \frac{1}{x_k}, k = 0,1,2,\cdots$$

所产生的迭代序列 $\{x_k\}$,对于 $x_0 \geqslant 1$ 都收敛于$\sqrt{2}$.

2. 用迭代法求方程 $e^x - 4x = 0$ 的实根,精确到三位有效数字.

3. 判断能否用迭代法求下列方程的数值解,如果不能,试将方程改写成能用迭代法求解的形式.

(1) $x = (\cos x + \sin x)/4$; (2) $x = 4 - 2^x$.

4. 求方程 $xe^x - 1 = 0$ 的一个实根,要求精度 $\varepsilon = 10^{-5}$.

5. 设近似值 $T_0 = S_0 = 35.70$ 具有四位有效数字,计算中无舍入误差,试分析分别用递推公式

$$T_{k+1} = 5T_k - 142.8 \text{ 和 } S_{k+1} = \frac{1}{5}S_k - 142.8$$

计算 T_{20} 和 S_{20} 所得结果是否可靠.

2.5 迭代过程的加速方法及其 MATLAB 程序

对于收敛的迭代过程,只要迭代足够多次,就可以使结果达到任意的精度,但有时迭代过程收敛缓慢,从而使计算量变得很大,因此,迭代过程的加速是个重要的问题.

2.5.1 加权迭代法

设 x_k 是根 x^* 的某个近似值,用迭代公式 $x_{k+1} = \varphi(x_k)$ $(k = 1,2,\cdots)$ 校正一次得

$$\bar{x}_{k+1} = \varphi(x_k),$$

而由微分中值定理,有

$$\bar{x}_{k+1} - x^* = \varphi'(\xi)(x_k - x^*) \ (\xi \text{ 在 } x_k \text{ 与 } x^* \text{ 之间}).$$

假设 $\varphi'(x)$ 在所考察的范围内改变不大,近似地取某个近似值为 L,则有

$$x_{k+1} - x^* \approx L(x_k - x^*), \tag{2.9}$$

得

$$x^* \approx \frac{1}{1-L}x_{k+1} - \frac{L}{1-L}x_k.$$

这就是说,若将迭代值 x_{k+1} 与 x_k 加权迭代平均,即取上式右端为

$$x_{k+1}=\frac{1}{1-L}\bar{x}_{k+1}-\frac{L}{1-L}x_k=\bar{x}_{k+1}+\frac{L}{1-L}(\bar{x}_{k+1}-x_k),$$

则 x_{k+1} 是比 $\bar{x}_{k+1}$ 更好的根的近似值.这样得到加权迭代的公式为

$$\begin{cases}\bar{x}_{k+1}=\varphi(x_k), & \text{迭代,}\\ x_{k+1}=\bar{x}_{k+1}+\dfrac{L}{1-L}(\bar{x}_{k+1}-x_k), & \text{改进}\end{cases}\quad(k=0,1,2,\cdots),\tag{2.10}$$

用加权迭代的公式(2.10)求方程根的近似值的方法称**加权迭代法**.

2.5.2 加权迭代法的 MATLAB 程序

(一) 加权迭代法的 MATLAB 主程序

加权迭代法的 MATLAB 主程序

已知初始值 x_0、精度 *tol* 和迭代公式 $x_{k+1}=\varphi(x_k)$,求满足精度 *tol* 的近似根 x_k 和它的函数值 y_k 及迭代次数 k.

根据加权迭代的公式(2.10)和已知条件,编写一个名为 jasudd.m 的 M 文件如下:

```
function [k,xk,yk] = jasudd (x0,tol,L,ddmax)
x1(1) = x0;
for i =1: ddmax
       x(i +1) = fun(x1(i));
       x1(i +1) = x(i +1) +L*( x(i +1) - x1(i))/(1 -L);
       piancha = abs(x1(i +1) -x1(i));
       xdpiancha = piancha/( abs(x1(i +1)) +eps);
       i = i +1;xk = x1(i);yk = fun(x1(i));[(i -1) xk yk]
       if(piancha < tol) |(xdpiancha < tol)
              k = i - 1; xk = x1(i);
              return;
       end
end
if i > ddmax
       disp('迭代次数超过给定的最大值 ddmax')
       k = i -1;xk = x1(i);[(i -1) xk yk];
       return;
end
P = [(i -1),xk,yk]';
```

(二) 使用加权迭代法的 MATLAB 程序求方程 $f(x)=0$ 的近似根(精度为

***tol*)的步骤**

步骤 1 把给定的方程 $f(x)=0$ 改写成容易迭代的形式 $x=\varphi(x)$,求 $L\approx\varphi'(x_0)$;

步骤 2 建立 M 文件 fun.m

```
function y = fun(x)
        y = φ(x);
```

步骤 3 保存名为 jasudd.m 的 M 文件;

步骤 4 在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk] = jasudd (x0,tol,L,ddmax),
```

其中输入的量:初始值 x_0,$L\approx\varphi'(x_0)$,允许最大迭代次数 *ddmax*(迭代次数超过给定的 *ddmax* 时,运行后输出信息:'迭代次数超过给定的最大值 ddmax')及精度 *tol* 都是具体给定的数值,运行后输出迭代次数 k、满足精度 *tol* 的近似根 x_k 和它的函数值 $y_k=\varphi(x_k)$(参见例 2.5.1).

例 2.5.1 用(2.10)式求 $2e^{-x}-x=0$ 在 0.85 附近的一个近似根,要求精度 $\varepsilon=10^{-6}$.

解 (1)取 $\varphi(x)=2e^{-x}$,$x_0=0.85$,则 $\varphi'(x_0)=-2e^{-x_0}\approx-0.855$,故(2.10)式的具体形式为

$$\begin{cases}\bar{x}_{k+1}=2e^{-x_k}, & \text{迭代},\\ x_{k+1}=\bar{x}_{k+1}+\dfrac{-0.855}{1+0.855}(x_{k+1}-x_k), & \text{改进}.\end{cases}$$

(2) 建立并保存名为 fun.m 的 M 文件

```
function y = fun(x)
        e = exp(1); y = 2*e^(-x);
```

(3) 保存名为 jasudd.m 的 M 文件.

(4) 在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk] = jasudd (0.85,1e-6,-0.855,100)
```

(5) 运行后输出结果

```
[k,xk,yk] =
     1.00000000000000   0.85260370028041   0.85260703830561
     2.00000000000000   0.85260549975491   0.85260550406236
     3.00000000000000   0.85260550207699   0.85260550208255
k =3; xk =0.852606; yk = 0.852606.
```

注意 例 2.4.7 迭代 55 次得到精度 $\varepsilon=10^{-6}$ 的根的近似值 $x_k=0.852\,606$,而这里只要迭代 3 次得出相同的结果,可见加权迭代法的效果是相当显著的.然而,此迭代法有个缺点,由于其中含有导数 $\varphi'(x)$ 的有关信息 L,实际使用不方便.下面的艾特肯加速方法改进了这个缺点.

2.5.3 艾特肯(Aitken)加速方法

将迭代值 $\bar{x}_{k+1}=\varphi(x_k)$ 再迭代一次,得

$$\tilde{x}_{k+1}=\varphi(\bar{x}_{k+1}).$$

而由微分中值定理,有

$$\tilde{x}_{k+1}-x^*=\varphi'(\xi)(x_{k+1}-x^*)\ (\xi\text{ 在 }x_{k+1}\text{与 }x^*\text{之间}).$$

假设 $\varphi'(x)$ 在所考察的范围内改变不大,近似的取某个近似值为 L,则有

$$\tilde{x}_{k+1}-x^*\approx L(\bar{x}_{k+1}-x^*).$$

将上式与式(2.9)联立,消去未知的 L,有

$$\frac{\bar{x}_{k+1}-x^*}{\tilde{x}_{k+1}-x^*}\approx\frac{x_k-x^*}{x_{k+1}-x^*},$$

由此得

$$x^*\approx\tilde{x}_{k+1}-\frac{(\tilde{x}_{k+1}-x_{k+1})^2}{\tilde{x}_{k+1}-2\bar{x}_{k+1}+x_k}.$$

取上式右端作为新的改进值,记作

$$x_{k+1}=\tilde{x}_{k+1}-\frac{(\tilde{x}_{k+1}-\bar{x}_{k+1})^2}{\tilde{x}_{k+1}-2\bar{x}_{k+1}+x_k},$$

这样得到加速迭代的公式不再含有导数的信息,但是它需要两次迭代.其具体计算公式为

$$\begin{cases}\bar{x}_{k+1}=\varphi(x_k), & \text{迭代},\\ \tilde{x}_{k+1}=\varphi(\bar{x}_{k+1}), & \text{迭代},\\ x_{k+1}=\tilde{x}_{k+1}-\dfrac{(\tilde{x}_{k+1}-x_{k+1})^2}{\tilde{x}_{k+1}-2\bar{x}_{k+1}+x_k}, & \text{改进},\end{cases}\tag{2.11}$$

用式(2.11)计算方程 $x=\varphi(x)$ 的近似根的方法称为**艾特肯加速方法**.

例 2.5.2 用艾特肯加速方法求方程 $f(x)=x^3+4x^2-10=0$ 在 $x_0=1.5$ 附近的近似根,取 $\varphi(x)=\sqrt{\dfrac{10}{4+x}}$,精确到 $|x_{k+1}-x_k|<10^{-5}$.

解 取 $\varphi(x)=\sqrt{\dfrac{10}{4+x}}$,根据艾特肯加速迭代公式得

$$\begin{cases}\bar{x}_{k+1}=\sqrt{\dfrac{10}{4+x_k}}, & \text{迭代},\\ \tilde{x}_{k+1}=\sqrt{\dfrac{10}{4+\bar{x}_{k+1}}}, & \text{迭代},\\ x_{k+1}=\tilde{x}_{k+1}-\dfrac{(\tilde{x}_{k+1}-\bar{x}_{k+1})^2}{\tilde{x}_{k+1}-2\bar{x}_{k+1}+x_k}, & \text{改进}.\end{cases}$$

取 $x_0=1.5$，当 $k=0$ 时，

$$\begin{cases} x_1=\sqrt{\dfrac{10}{4+x_0}}=\sqrt{\dfrac{10}{4+1.5}}=1.348\ 400, \\ \tilde{x}_1=\sqrt{\dfrac{10}{4+x_1}}=\sqrt{\dfrac{10}{4+1.348\ 400}}=1.367\ 376, \\ x_1=\tilde{x}_1-\dfrac{(\tilde{x}_1-\bar{x}_1)^2}{\tilde{x}_1-2\bar{x}_1+x_0}=1.365\ 265. \end{cases}$$

当 $k=1$ 时，

$$\begin{cases} \bar{x}_2=\sqrt{\dfrac{10}{4+x_1}}=\sqrt{\dfrac{10}{4+1.365\ 265}}=1.365\ 226, \\ \tilde{x}_2=\sqrt{\dfrac{10}{4+\bar{x}_2}}=\sqrt{\dfrac{10}{4+1.365\ 226}}=1.365\ 231, \\ x_2=\tilde{x}_2-\dfrac{(\tilde{x}_2-\bar{x}_2)^2}{\tilde{x}_2-2x_2+x_1}=1.365\ 230. \end{cases}$$

当 $k=2$ 时，

$$\begin{cases} x_3=\sqrt{\dfrac{10}{4+x_2}}=\sqrt{\dfrac{10}{4+1.365\ 230}}=1.365\ 230, \\ \tilde{x}_3=\sqrt{\dfrac{10}{4+x_3}}=\sqrt{\dfrac{10}{4+1.365\ 230}}=1.365\ 230, \\ x_3=\tilde{x}_3-\dfrac{(\tilde{x}_3-\bar{x}_3)^2}{\tilde{x}_3-2x_3+x_2}=1.365\ 230. \end{cases}$$

因为 $|x_3-x_2|<10^{-5}$，所以用艾特肯加速方法求方程 $f(x)=0$ 在 $x_0=1.5$ 附近的近似根为 $x_3=1.365\ 23$.

2.5.4 艾特肯加速方法的 MATLAB 程序

(一) 艾特肯加速方法的 MATLAB 程序

> **艾特肯加速方法的 MATLAB 程序**
>
> 已知初始值 x_0、精度 *tol* 和迭代公式 $x_{k+1}=\varphi(x_k)$，求满足精度 *tol* 的近似根 x_k 和它的函数值 y_k 及迭代次数 k，$p=[k',x1',x2',x']$.

根据艾特肯加速法的公式(2.11)和已知条件，编写一个名为 Aitken.m 的 M 文件如下：

```
function [k,xk,yk,p] = Aitken (x0,tol, ddmax)
x(1) =x0;
for i =1: ddmax
```

```
        x1(i +1) = fun(x(i)); x2(i +1) = fun(x1(i +1));
        x(i +1) =x2(i +1) -(x2(i +1) -x1(i +1))^2/(x2(i +1) -2*x1(i +1) + x(i));
        piancha = abs(x(i +1) - x(i));
         xdpiancha = piancha/( abs(x(i +1)) +eps);
        i = i +1 ; xk = x(i);yk = fun(x(i));
                if(piancha < tol) |(xdpiancha < tol)
                    k = i -1; xk = x(i);yk = fun(x(i));m = [0,1: i -1];
                    p = [m',x1',x2',x'];
                       return;
                end
    end
    if i > ddmax
          disp('迭代次数超过给定的最大值 ddmax')
          k = i -1;xk =x(i);yk = fun(x(i));m = [0,1: i -1];p = [m',x1',x2',x']
          return;
    end
    m = [0,1:i -1]; p = [m',x1',x2',x'];
```

(二) 使用艾特肯加速方法的 MATLAB 程序求方程 $f(x)=0$ 的近似根(精度为 tol)的步骤

步骤 1 把给定的方程 $f(x)=0$ 改写成容易迭代的形式 $x=\varphi(x)$.

步骤 2 建立 M 文件 fun.m

```
function y = fun(x)
         y = φ(x);
```

步骤 3 保存名为 Aitken.m 的 M 文件;

步骤 4 在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk,p] = Aitken (x0,tol, ddmax)
```

其中输入的量:初始值 x_0、允许最大迭代次数 $ddmax$ (迭代次数超过 $ddmax$ 时,运行后输出信息:'迭代次数超过给定的最大值 ddmax')及精度 tol 都是具体给定的数值,运行后输出迭代次数 k、满足精度 tol 的近似根 x_k 和它的函数值 $y_k=\varphi(x_k)$ 的值(参见例 2.5.3).

例 2.5.3 用艾特肯加速方法求 $2e^{-x}-x=0$ 在 0.85 附近的一个近似根,要求精度 $\varepsilon=10^{-6}$.

解 (1) 取 $\varphi(x)=2e^{-x}$, $x_0=0.85$.

(2) 建立 M 文件 fun.m

```
function y = fun(x)
         x1 = exp(1); y =2*x1^( -x);
```

(3) 保存名为 Aitken.m 的 M 文件.

(4) 在 MATLAB 工作窗口输入程序

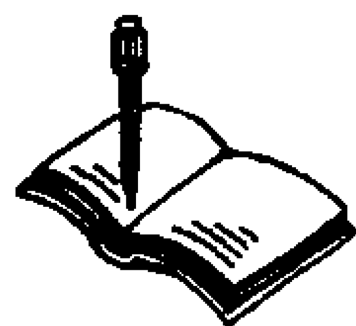
```
>> [k,xk,yk,p] = Aitken(0.85,1e-6, 100)
```

(5) 运行后输出结果

```
k =3,xk =0.85260550201343,yk =0.85260550201373
p =                0                 0                  0  0.85000000000000
  1.00000000000000  0.85482986389745  0.85071110652484  0.85260683568607
  2.00000000000000  0.85260436491811  0.85260647150826  0.85260550201407
  3.00000000000000  0.85260550201343  0.85260550201398  0.85260550201373
```

注意 (1) 例2.5.1与例2.5.3比较可知,艾特肯加速方法和加权迭代法都迭代3次得到精度 $\varepsilon=10^{-6}$的根的近似值 $x_k=0.852\ 606$ 相同.

(2) 用命令 x = solve('2*2.718281828^(-x) - x = 0'),求得方程 $2e^{-x}-x=0$ 的根为 $x=0.852\ 605\ 502\ 079\ 988\ 982\ 669\ 281\ 424\ 164\ 67$,将迭代3次的艾特肯加速方法得到根的近似值0.852 605 502 079 99与加权迭代法得的结果0.852 605 502 076 99与根 x 比较可看出,前者比后者的精度高.



习 题 2.5

1. 分别用迭代法和艾特肯加速方法求方程 $xe^x=1$ 在0.5附近的近似根和它们的迭代次数,精确到 $\varepsilon=10^{-5}$.

2. 分别用加权迭代法和艾特肯加速方法求方程 $x^3-x-1=0$,在给定区间[1,2]上的近似根,精确到 $\varepsilon=10^{-4}$.

3. 用艾特肯加速方法求方程 $x-2^{-x}=0$,在给定区间[0,1]上的近似根,精确到 $\varepsilon=10^{-4}$.

4. 分别用加权迭代法和艾特肯加速方法求 $f(x)=x^3+4x^2-10=0$ 在 $x_0=1.5$ 附近的近似根,取 $\varphi(x)=\sqrt{\dfrac{10}{4+x}}$,精度到 $|x_{k+1}-x_k|<10^{-5}$,再用二分法求区间[1,2]内的近似根,并将三种方法的计算结果进行比较.

2.6 牛顿(Newton)切线法及其 MATLAB 程序

2.6.1 牛顿切线法

这是一种具体的构造迭代公式的方法.

(一) 牛顿迭代公式

设方程 $f(x)=0$ 的根 x^* 的近似根是 x_k,则 $f(x)$ 在点 x_k 作泰勒(Taylor)展开,去掉2阶及2阶以上项(即线性化)后得

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k).$$

设 $f'(x_k) \neq 0$，用 x_{k+1} 代替右端的 x，由 $f(x)=0$ 就得到迭代公式

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (k=0,1,2,\cdots). \tag{2.12}$$

这相当于选择迭代函数为

$$\varphi(x) = x - \frac{f(x)}{f'(x)}, \tag{2.13}$$

称迭代公式(2.12)为**牛顿迭代公式**(简称**牛顿公式**)，它是局部收敛的. 用牛顿公式求方程根的方法称为**牛顿切线法**.

(二) 牛顿公式的几何意义

如果把牛顿公式(2.12)的左端的 x_{k+1} 用 Y_k 代替，得

$$Y_k = f(x_k) + f'(x_k)(x - x_k) \quad (k=0,1,2,\cdots)$$

表示曲线 $y=f(x)$ 在点 $(x_k, f(x_k))$ 处的切线. 所以，牛顿公式的几何意义如图 2－16 所示. 设 x_0 是方程 $f(x)=0$ 根的一个近似值，过点 $(x_0, f(x_0))$ 作曲线 $y=f(x)$ 的切线 MN，它与 x 轴交于点 x_1；x_1 作为根的新的近似值，过点 $(x_1, f(x_1))$ 作曲线 $y=f(x)$ 的切线 M_1N_1，它与 x 轴的交点即为 x_2；x_2 作为根的新的近似值，如此继续下去得到序列 $x_1, x_2, \cdots$. 当此点列 $\{x_k\}$ 收敛到方程 $f(x)=0$ 的根 x^* 时，可求出满足精度要求的近似根 x_k(见图 2－16). 这就是牛顿切线法名称的由来，可以看出它是**线性化与迭代法的结合**.

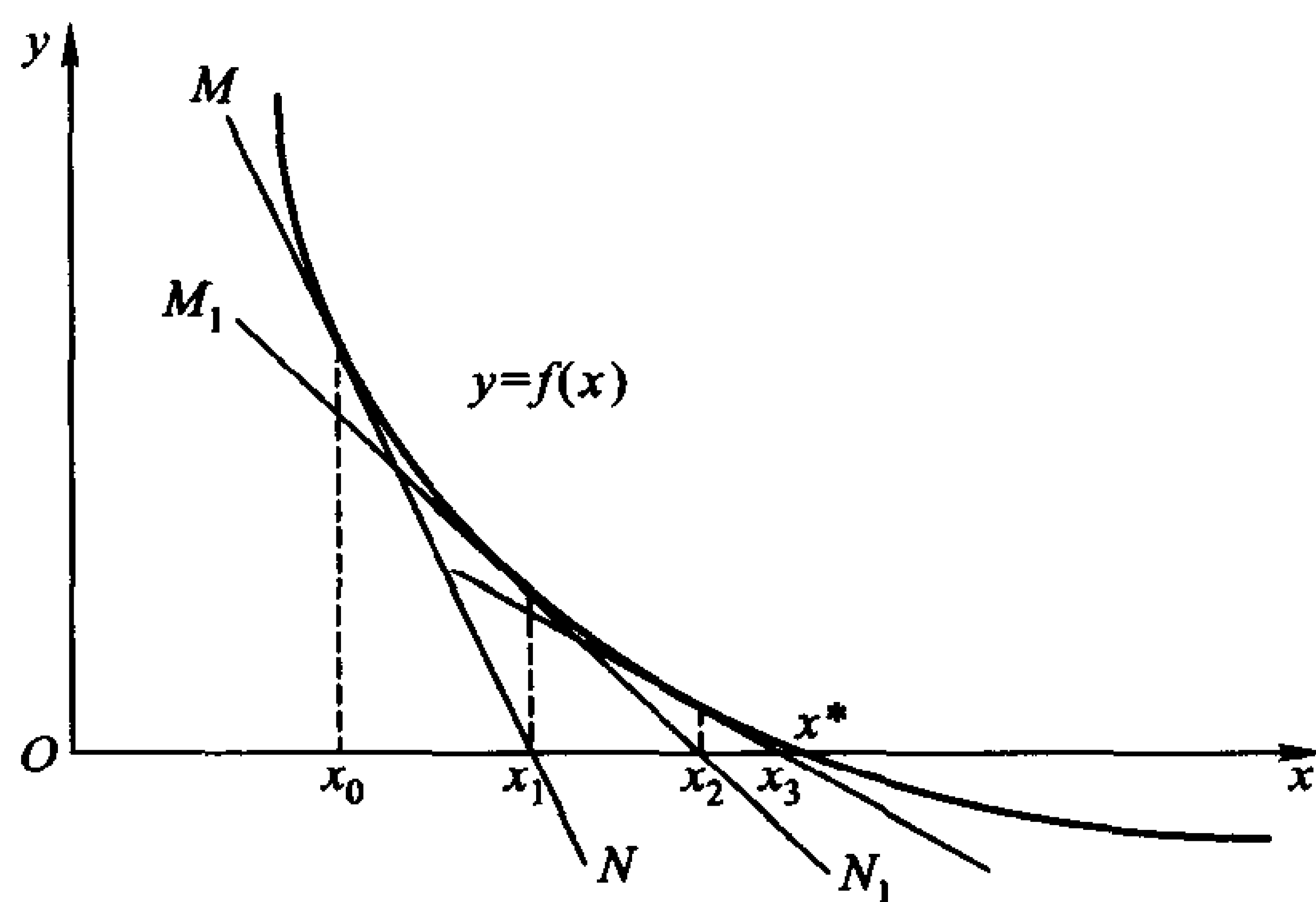


图 2－16 牛顿切线法的几何意义

2.6.2 牛顿切线法的收敛性及其 MATLAB 程序

(一) 迭代法的收敛性

为进一步研究收敛速度问题，引入阶的概念.

定义 2.2 设由迭代公式 $x_{k+1}=\varphi(x_k)$ 得到的迭代序列 $\{x_k\}$ 收敛于方程 $x=\varphi(x)$ 的根 x^*，记 $e_k=x_k-x^*$，若

$$\lim_{k\to\infty}\frac{e_{k+1}}{e_k^p}=c\neq 0\ (\text{其中 } p \text{ 为一正数}) \tag{2.14}$$

称序列 $\{x_k\}$ 是 p **阶收敛**. 特别地，$p=1$ 时称**线性收敛**，$p>1$ 时称**超线性收敛**，$p=2$时称**平方收敛**. 显然，p 越大收敛速度越快.

定理 2.5 设由迭代公式 $x_{k+1}=\varphi(x_k)$ 得到迭代序列 $\{x_k\}$，若 $\varphi^{(p)}(x)$ 在所求的根 x^* 的某个邻域内 $U(x^*)$ 连续，且

$$\varphi'(x^*)=\cdots=\varphi^{(p-1)}(x^*)=0,\varphi^{(p)}(x^*)\neq 0, \tag{2.15}$$

则迭代序列 $\{x_k\}$ 在所求的根 x^* 的邻域 $U(x^*)$ 内是 p **阶收敛**.

证明 利用 $\varphi(x_k)$ 在 x^* 的泰勒展开

$$\varphi(x_k)=\varphi(x^*)+\varphi'(x^*)(x_k-x^*)+\cdots+\frac{\varphi^{(p)}(x^*)}{p!}(x_k-x^*)^p+\cdots, \tag{2.16}$$

并注意到 $x_{k+1}=\varphi(x_k)$，$x^*=\varphi(x^*)$ 和 $e_k=x_k-x^*$，由(2.16)式得

$$e_{k+1}=\varphi'(x^*)e_k+\cdots+\frac{\varphi^{(p)}(x^*)}{p!}e_k^p+\cdots, \tag{2.17}$$

于是根据阶的定义中(2.14)式，若 $\varphi'(x^*)\neq 0$，则 $\{x_k\}$ 是 1 阶收敛(线性收敛)；若 $\varphi'(x^*)=\cdots=\varphi^{(p-1)}(x^*)=0,\varphi^{(p)}(x^*)\neq 0$，则 $\{x_k\}$ p 阶收敛. 证毕.

例 2.6.1 分别判别例 2.4.1 中下面的两个迭代公式对于在单根 $x^*=\sqrt{11}-1$附近的任意初始值的收敛速度.

(1) $x_{k+1}=10/(x_k+2)$；(2) $x_{k+1}=x_k-(x_k^2+2x_k-10)/(2x_k+2)$.

解 (1) 因为对于迭代公式(1)对应的迭代函数 $x=\varphi_2(x)=10/(x+2)$，$\varphi_2'(x)=-10/(x+2)^2$，$\varphi_2'(x^*)\neq 0$，所以，由定理 2.6.1 知，对于在单根 x^* 附近的任意初始值由迭代公式(1)生成的迭代序列 $\{x_k\}$ 都是 1 阶收敛(线性收敛).

(2) 对于收敛很快的牛顿切线法的迭代函数

$$x=\varphi_3(x)=x-(x^2+2x-10)/(2x+2),$$

可以算出 $\varphi_3'(x^*)=0,\varphi_3''(x^*)\neq 0$，故 $\{x_k\}$ 是**平方收敛**.

由此可知，如何选择$\varphi(x)$构造迭代公式，对于收敛速度是至关重要的.

(二) 牛顿切线法的局部收敛性

定理 2.6 设 x^* 是方程 $f(x)=0$ 的根，函数 $f(x)$ 在 x^* 的某一邻域 $U(x^*,\delta)$ (其中 $0<\delta<1$) 内具有二阶连续导数，迭代函数为 $\varphi(x)=x-\dfrac{f(x)}{f'(x)}$，当 $x\in U(x^*,\delta)$ 时，$\varphi(x)\in U(x^*,\delta)$. 对于 $x_0\in U(x^*,\delta)$，牛顿迭代公式(2.12)产生的迭代序列为 $\{x_k\}$.

(1) 如果

$$|\varphi'(x^*)| = \left|\frac{f(x^*)f''(x^*)}{[f'(x^*)]^2}\right| < 1, \tag{2.18}$$

那么迭代序列$\{x_k\}$收敛于x^*.

(2) 如果x^*是$f(x)=0$的单根,那么迭代序列$\{x_k\}$在x^*附近是二阶收敛的,即平方收敛. 且

$$\lim_{k\to\infty}\frac{x_{k+1}-x^*}{(x_k-x^*)^2}=\frac{f''(x^*)}{2f'(x^*)}.$$

(3) 如果x^*是$f(x)=0$的二重根,那么迭代序列$\{x_k\}$在x^*附近是一阶收敛,即线性收敛的,且重数越高收敛越慢.

证明 (1) 因为$\varphi(x)=x-\dfrac{f(x)}{f'(x)}$,所以$\varphi'(x^*)=\dfrac{f(x^*)f''(x^*)}{[f'(x^*)]^2}$,$\varphi''(x^*)=\dfrac{f''(x^*)}{f'(x^*)}$. 根据定理 2.4.1 知(1)显然成立.

(2) 若x^*是$f(x)=0$的单根,即$f(x^*)=0, f'(x^*)\neq 0, f''(x^*)\neq 0$ 则$\varphi'(x^*)=0, \varphi''(x^*)\neq 0$. 这时,由定理 2.6.1 可知,迭代序列$\{x_k\}$在$x^*$附近是平方收敛的.

由泰勒公式

$$0=f(x^*)=f(x_k)+f'(x_k)(x^*-x_k)+\frac{f''(\xi)}{2!}(x^*-x_k)^2,$$

其中ξ在x^*与x_k之间.

$$-f(x_k)=f'(x_k)(x^*-x_k)+\frac{f''(\xi)}{2!}(x^*-x_k)^2,$$

$$x_k-\frac{f(x_k)}{f'(x_k)}-x^*=\frac{f''(\xi)}{2f'(x_k)}(x^*-x_k)^2,$$

$$x_{k+1}-x^*=\frac{f''(\xi)}{2f'(x_k)}(x^*-x_k)^2,$$

$$\lim_{k\to\infty}\frac{x_{k+1}-x^*}{(x^*-x_k)^2}=\lim_{k\to\infty}\frac{f''(\xi)}{2f'(x_k)}=\frac{f''(x^*)}{2f'(x^*)}.$$

(3) 当x^*是$f(x)=0$的二重根时,$\varphi'(x^*)\neq 0$,根据定理 2.6.1,不难看出(3)显然成立.

说明 (1) 牛顿切线法局部收敛的充分条件是$|\varphi'(x^*)|<1$.

(2) 若x^*是$f(x)=0$的单根,则牛顿切线法收敛速度很快. 若x^*是二重根时,牛顿切线法收敛速度很慢,且重数越高收敛越慢.

例 2.4.1 中收敛很快的迭代公式$x_{k+1}=x_k-(x_k^2+x_k-14)/(2x_k+1)$,正是用的牛顿切线法.

(三) 判别牛顿切线法的局部收敛性的 MATLAB 程序

判别牛顿切线法的局部收敛性的 MATLAB 主程序

根据牛顿切线法局部收敛的条件 $|\varphi'(x^*)| = \left|\frac{f(x^*)f''(x^*)}{[f'(x^*)]^2}\right| < 1.$

输入的量:x 是方程 $f(x)=0$ 的根 x^* 或满足 $|x_0 - x^*| << 1$ 的初始值 x_0,函数 $f(x)$ 及其一阶导数 $f'(x)$ 和二阶导数 $f''(x)$;

输出的量:$|\varphi'(x^*)|$ 或 $|\varphi'(x_0)|$ 的值. 如果 $|\varphi'(x^*)| < 1$ 或 $|\varphi'(x_0)| < 1$ 时,运行后输出信息'恭喜您! 此迭代序列收敛,ϕ(x)的导数值的绝对值 y = |dϕ(x)/dx |和方程f(x) =0的函数 f(x)的值 f 如下';否则,运行后输出信息'请注意观察下面显示的 ϕ(x)的导数值的绝对值 y = |dϕ(x)/dx |和方程 f(x) =0 的函数 f(x)的值 f '.

根据牛顿切线法局部收敛的条件和方程 $f(x)=0$ 的函数 $f(x)$ 及其一、二阶导数,编写程序:

```
function [y,f] = newjushou(x)
f = fnq(x); fz = fnq(x) * ddfnq(x)/((dfnq(x))^2 + eps);
y = abs(fz);
if (y<1)
    disp('恭喜您! 此迭代序列收敛,ϕ(x)导数值的绝对值 y = |dϕ
    (x)/dx |和方程 f(x) =0 的函数 f(x)的值 f 如下')
else
    disp('请注意观察下面显示的 ϕ(x)的导数值的绝对值 y = |dϕ(x)/dx |
    和方程 f(x) =0 的函数 f(x)的值 f ')
end
P = [y,f]';
```

(四) 选取初始值 x_0 的方法

对于一般的方程 $f(x)=0$ 来说,人们很难求出它的全部精确解,所以牛顿切线法局部收敛的条件(2.18)很少被直接应用. 人们最关心的问题是:在不知道方程 $f(x)=0$ 根的精确值的情况下,如何取初始值 x_0,使式(2.12)产生的迭代序列 $\{x_k\}$ 收敛到要求的方程根. 这个问题我们可以用下面的方法解决:

(1) 首先用作图法画出函数 $f(x)$ 的图形,根据图确定 $f(x)=0$ 的根 x^* 的位置,并取初始值 x_0,满足 x_0 与 x^* 的距离 $|x_0 - x^*| < 1$. 当然,$|x_0 - x^*|$ 越小越好.

(2) 用牛顿切线法的局部收敛性的 MATLAB 主程序判别由初始值 x_0 和式(2.12)产生的迭代序列 $\{x_k\}$ 的敛散性,如果 x_0 满足

$$0 < |\varphi'(x_0)| = \left|\frac{f(x_0)f''(x_0)}{[f'(x_0)]^2}\right| < 1, \tag{2.19}$$

那么,一般情况下,此迭代序列$\{x_k\}$收敛到方程$f(x)=0$的根,且此根是与初始值x_0距离最近的根.

说明 $|x_0-x^*|<1$且$0<|\varphi'(x_0)|<1$是牛顿切线法局部收敛的充分条件.否则,产生的迭代序列$\{x_k\}$可能不收敛到方程$f(x)=0$与初始值x_0距离最近的根,甚至发散,见下面几种情况:

(1) 如果$0<|\varphi'(x_0)|<1$且$|x_0-x^*|\geqslant 1$,则此迭代序列$\{x_k\}$收敛的根不一定是与初始值x_0距离最近的根(见例 2.6.6 中$x_0=8$).

(2) 如果$\varphi'(x_0)=0$且$|x_0-x^*|\neq 0$,则此迭代序列$\{x_k\}$不一定收敛(见 2.6.4 目牛顿切线法的缺陷中(2)的举例).

(3) 如果$|\varphi'(x_0)|>1$且$|x_0-x^*|<1$时,这个初始值所产生的迭代数列$\{x_k\}$有时也收敛到离它最近的根,即$\{x_k\}$是局部收敛的(见后面的例 2.6.6 中$x_0=2$).这说明$|\varphi'(x_0)|<1$是充分条件,非必要条件.

例 2.6.2 用牛顿切线法的局部收敛性判别方程$e^x\sin x=4$的近似根时,由下列初始值x_0产生的迭代序列是否收敛?

(1) $x_0=-1$; (2) $x_0=0$; (3) $x_0=1$; (4) $x_0=2$; (5) $x_0=5.5$; (6) $x_0=8$.

解 (1)保存名为 newjushou.m 的 M 文件.

建立并保存名为 fnq.m 的 M 文件

```
function f = fnq(x)
            e = exp(1); f = 4 - e^x * sin(x);
```

建立并保存名为 dfnq.m 的 M 文件

```
function dy = dfnq(x)
            e = exp(1);  dy = - e^x * (sin(x) + cos(x));
```

建立并保存名为 ddfnq.m 的 M 文件

```
function ddy = ddfnq(x)
            e = exp(1);  ddy = -2 * e^x * cos(x);
```

在 MATLAB 工作窗口输入程序

```
>> [y,f] = newjushou( -1)
```

运行后输出结果

```
请注意观察下面显示的 φ(x)的导数值的绝对值 y = |dφ(x)/dx|和方程 f(x) = 0 的函数 f(x)的值 f
    y =
       139.5644
    f =
         4.3096
```

说明 实际上,这个初始值所产生的迭代序列$\{x_k\}$发散,见例 2.6.6.

(2) 输入程序

```
>> [y,f] = newjushou(0)
```

运行后输出结果

```
请注意观察下面显示的 φ(x)的导数值的绝对值 y = |dφ(x)/dx|和方程 f(x) = 0的函数 f(x)的值 f
    y =
        8.0000
    f =
        4
```

说明 实际上，这个初始值所产生的迭代序列$\{x_k\}$收敛到方程的根 2.925 32，而不是收敛到离它最近的根 1.400 81，即不是局部收敛的，见例 2.6.6.

(3) 输入程序

```
>> [y,f] = newjushou(1)
```

运行后输出结果

```
恭喜您！此迭代序列收敛，φ(x)导数值的绝对值 y = |dφ(x)/dx |和方程 f(x) = 0的函数 f(x)的值 f 如下
    y =
      0.3566
    f =
      1.7126
```

说明 实际上，这个初始值所产生的迭代序列$\{x_k\}$收敛到离它最近的根 1.400 81，即$\{x_k\}$是局部收敛的，见例 2.6.6.

(4) 输入程序：

```
>> [y,f] = newjushou(2)
```

运行后输出结果

```
请注意观察下面显示的 φ(x)的导数值的绝对值 y = |dφ(x)/dx |和方程 f(x) = 0的函数 f(x)的值 f
    y =
      1.2593
    f -
      -2.7188
```

说明 虽然 $y = 1.259\,3 > 1$，不满足牛顿切线法局部收敛的条件$|\varphi'(x^*)| < 1$. 但是，实际上，这个初始值所产生的迭代序列$\{x_k\}$收敛到离它最近的根 1.400 81，即$\{x_k\}$是局部收敛的. 这说明$|\varphi'(x_0)| < 1$ 是充分条件，非必要条件(见例 2.6.6).

(5) 输入程序

```
>> [y,f] = newjushou(5.5)
```

运行后输出结果

```
请注意观察下面显示的φ(x)的导数值的绝对值 y = |dφ(x)/dx|和方程f(x) = 0的函数 f(x)的值 f
y =
   1.0447e+005
f =
   176.6400
```

说明 实际上,这个初始值所产生的迭代序列$\{x_k\}$收敛到235.619,但不是此方程的根,比较初始值5.5与迭代值235.619,可见$\{x_k\}$不是局部收敛的,见例2.6.6.

(6) 输入程序

```
>> [y,f] = newjushou(8)
```

运行后输出结果

```
恭喜您! 此迭代序列收敛,φ(x)导数值的绝对值 y = |dφ(x)/dx|和方程f(x) = 0的函数 f(x)的值 f 如下
y =
   0.4038
f =
   -2.9452e+003
```

说明 实际上,这个初始值所产生的迭代序列$\{x_k\}$收敛到方程的根6.290 60,而不是收敛到离它最近的根9.424 46,即不是局部收敛的,见例2.6.6.

2.6.3 牛顿切线法的 MATLAB 程序

(一) 用牛顿迭代公式(2.12)求方程$f(x)=0$的近似根x_k的一般步骤

步骤1 选取初始值x_0和迭代的最大次数 *gxmax* 以及精度ε,计算$f(x_0)$.

步骤2 用牛顿迭代公式(2.12)计算x_1,并求出$f(x_1)$.

步骤3 判断.

① 如果$|f(x_1)|<\varepsilon$,且$|x_1-x_0|<\varepsilon$或$|x_1-x_0|/|x_1|<\varepsilon$(其中$\varepsilon$是预先给定的精度),则迭代停止;否则,将$x_1$和$f(x_1)$代入牛顿迭代公式(2.12),重复步骤2和步骤3.

② 如果迭代的次数超过预先给定的最大次数 *gxmax*,则迭代停止,屏幕显示提示:'请注意:迭代次数超过给定的最大值 gxmax'.

(二) 牛顿切线法的 MATLAB 主程序

牛顿切线法求方程$f(x)=0$的近似根x_k(精度为 *tol*)需要自行编制程序.

牛顿切线法的 MATLAB 主程序

输入的量:初始值 x_0,近似根 x_k 的误差限 *tol*,近似根 x_k 的函数值 $f(x_k)$ 的误差限 *ftol*,迭代次数的最大值 *gxmax*、函数 $fnq(x)=f(x)$ 及其导数 $dfnq(x)=f'(x)$.

输出的量:迭代序列 $\{x_k\}$、迭代 k 次得到的迭代值 x_k(迭代次数超过 *gxmax* 时,运行后输出信息:'请注意:迭代次数超过给定的最大值 gxmax')、相邻两次迭代的偏差 $piancha=|x_k-x_{k-1}|$ 和它的偏差的相对误差 $xdpiancha=|x_k-x_{k-1}|/|x_k|$ 的值.

根据式(2.12)和已知条件编写一个名为 newtonqx.m 的 M 文件:

```
function [k,xk,yk,piancha,xdpiancha] = newtonqx(x0,tol,ftol,gxmax)
      x(1) = x0;
      for i = 1: gxmax
          x(i +1) =x(i) - fnq(x(i))/(dfnq(x(i)) +eps); piancha =abs(x(i +1) -x(i));
          xdpiancha = piancha/( abs(x(i +1)) + eps); i = i +1;
          xk = x(i);yk = fnq(x(i)); [(i -1) xk yk piancha xdpiancha]
        if (abs(yk) < ftol)&((piancha < tol) |(xdpiancha < tol))
              k = i -1; xk = x(i);[(i -1) xk yk piancha xdpiancha];
              return;
        end
      end
       if i > gxmax
            disp('请注意:迭代次数超过给定的最大值 gxmax。')
            k = i -1;xk = x(i):[(i -1)xk yk piancha xdpiancha];
            return;
      end
       [(i -1),xk,yk,piancha,xdpiancha]';
```

(三) 用牛顿切线法的 MATLAB 程序求方程 $f(x)=0$ 根的近似值(精度为 ε)的步骤

步骤 1 保存牛顿切线法的 MATLAB 主程序为 M 文件,命名为 newtonqx.m.

步骤 2 建立名为 fnq.m 的 M 文件

```
function y = fnq(x)
          y =f(x);
```

步骤 3 建立名为 dfnq.m 的 M 文件

```
function y = dfnq(x)
          y =f'(x);
```

步骤 4 在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk,piancha,xdpiancha] = newtonqx(x0,tol, ftol, gxmax)
```

步骤 5 运行后输出结果.

(四) 应用举例

例 2.6.3 用牛顿切线法求方程 $2x^3-3x^2+1=0$ 在 $x_0=-0.4$ 和 0.9 附近的近似根,要求精度 $\varepsilon=10^{-3}$. 然后判断此方法分别在方程的根 $x^*=-0.5$ 和 1 处的收敛速度.

解 (1) 先求方程的近似根.

① 保存牛顿切线法的 MATLAB 程序为 M 文件,命名为 newtonqx.m.

② 建立名为 fnq.m 的 M 文件

```
function y = fnq(x)
        y = 2 * x^3 - 3 * x^2 + 1;
```

③ 建立名为 dfnq.m 的 M 文件

```
function y = dfnq(x)
        y = 6 * x^2 - 6 * x;
```

④ 在MATLAB 工作窗口输入程序

```
>> [k,xk,yk,piancha,xdpiancha] = newtonqx( -0.4,0.001, 0.001,100)
```

⑤ 运行后输出初始值 $x_0=-0.4$ 的迭代结果如表 2-10 所示.

表 2-10

k	x_k	y_k	$\|x_k-x_{k-1}\|$	$\|x_k-x_{k-1}\|/\|x_k\|$
1	-0.516 7	-0.076 7	0.116 7	0.225 8
2	-0.500 4	-0.001 6	0.016 3	0.032 6
3	-0.500 0	-0.000 0	0.000 4	0.000 7

迭代次数 $k=3$,根的近似值 $x_k=-0.500$,它的函数值 $y_k=-1.759e-013$,偏差 *piancha* $=1.712e-007$ 和相对误差 *xdpiancha* $=3.423e-007$.

如果将步骤④命令中的 -0.4 改作 0.9,则运行后输出初始值 $x_0=0.9$ 的迭代结果为

k $=7$;*piancha* $=7.290e-004$;*xdpiancha* $=7.295e-004$;*xk* $=0.999$;

yk $=1.590e-006$.

(2) 再讨论收敛速度

比较初始值分别是 $x_0=-0.4$ 和 0.9 的两组结果,在 $x_0=-0.4$ 处迭代 3 次就得到单根 $x^* \doteq -0.5$ 的近似值 $x_k=-0.500$;而在 $x_0=0.9$ 处迭代 7 次才得到二重根 $x^*=1$ 的近似值 $x_k=0.999$. 可见,牛顿切线迭代法在单根处的迭代速度比二重根处的迭代速度快很多. 这正如前面讨论的结果一样,即若 x^* 是 $f(x)=0$

的单根,则牛顿切线法是平方收敛的;若 x^* 是 $f(x)=0$ 的二重根,则牛顿切线法是线性收敛的;我们也可以用阶的定义判断它们的收敛速度. 留给读者证明.

例 2.6.4 在传染病蔓延过程中,当已感染者(病人)与易感染者(健康人)进行接触时,健康人会变为病人. 假设在一个封闭的环境中总人数 N 不变,记 t 时刻健康人和病人在总人数中的比例分别为 $x(t)$ 和 $y(t)$. 设每个病人每天接触的平均人数为 a (称**日接触率**),于是其中有 ax 个健康人变为病人,yN 个病人每天使健康人减少 $axyN$ 个,所以 $x(t)$ 应满足微分方程

$$x'(t) = -axy. \tag{2.20}$$

再假设病人每天被治愈的比例为 b (称**日治愈率**), 即每天治愈的病人数为 byN. 从增加的 $axyN$ 中扣除减少的 byN,就是每天病人数的变化量,于是 $y(t)$ 满足微分方程

$$y'(t) = axy - by. \tag{2.21}$$

对于伤风、痢疾等痊愈后无免疫性的传染病,病人愈后又成为易感染者(健康人),于是总有 $x(t)+y(t)=1$,将此式代入方程(2.20),(2.21)得到关于 $x(t)$,$y(t)$的两个独立的方程,很容易地分别求解. 我们不再讨论这种情况.

对于天花、麻疹等痊愈后有**免疫性**的传染病,病人愈后不会成为易感染者(健康人),于是 $x(t)+y(t)\neq 1$,这时 $x(t)$,$y(t)$是两个独立变量,方程(2.20)、(2.21)要联立求解. 但是由于无法求出 $x(t)$,$y(t)$的解析解,只能从(2.20)、(2.21)消去 $\mathrm{d}t$ 得到

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{1}{\sigma x} - 1, \sigma = \frac{a}{b}. \tag{2.22}$$

因为 $\frac{1}{b}$ 表示传染期的平均天数,所以方程(2.22)中 σ 的含义是一个传染期内每个病人接触的平均人数,称**接触数**. 记初始时刻健康人和病人在总人数中的比例分别为 x_0,y_0,由方程(2.22)不难解出 $y=y(x)$.

通常人们关心的是,有多大比例的健康人不被感染成为病人. 这可以用传染过程结束($t\to\infty$)时健康人比例(记作 x_∞)来度量(可以证明 $t\to\infty$ 时病人比例 $y_\infty=0$).

(1) 日接触率 $a=1$,日治愈率 $b=0.3$,$x_0=0.98$,$y_0=0.02$,求 x_∞.

(2) 提高卫生水平使日接触率降低为 $a=0.6$ 时,结果如何. 若进一步改进医疗条件使日治愈率提高到 $b=0.5$,结果又如何.

(3) 如果在传染病蔓延前采取接种疫苗的办法,使初始时刻易感染者(健康人)的比例降低为 $x_0=0.70$ ($y_0=0.02$ 不变),结果会发生什么变化.

解 方程(2.22)在初始条件 $y(x_0)=y_0$ 下的解为

$$y = \frac{1}{\sigma}\ln\frac{x}{x_0} - x + (x_0 + y_0) \tag{2.23}$$

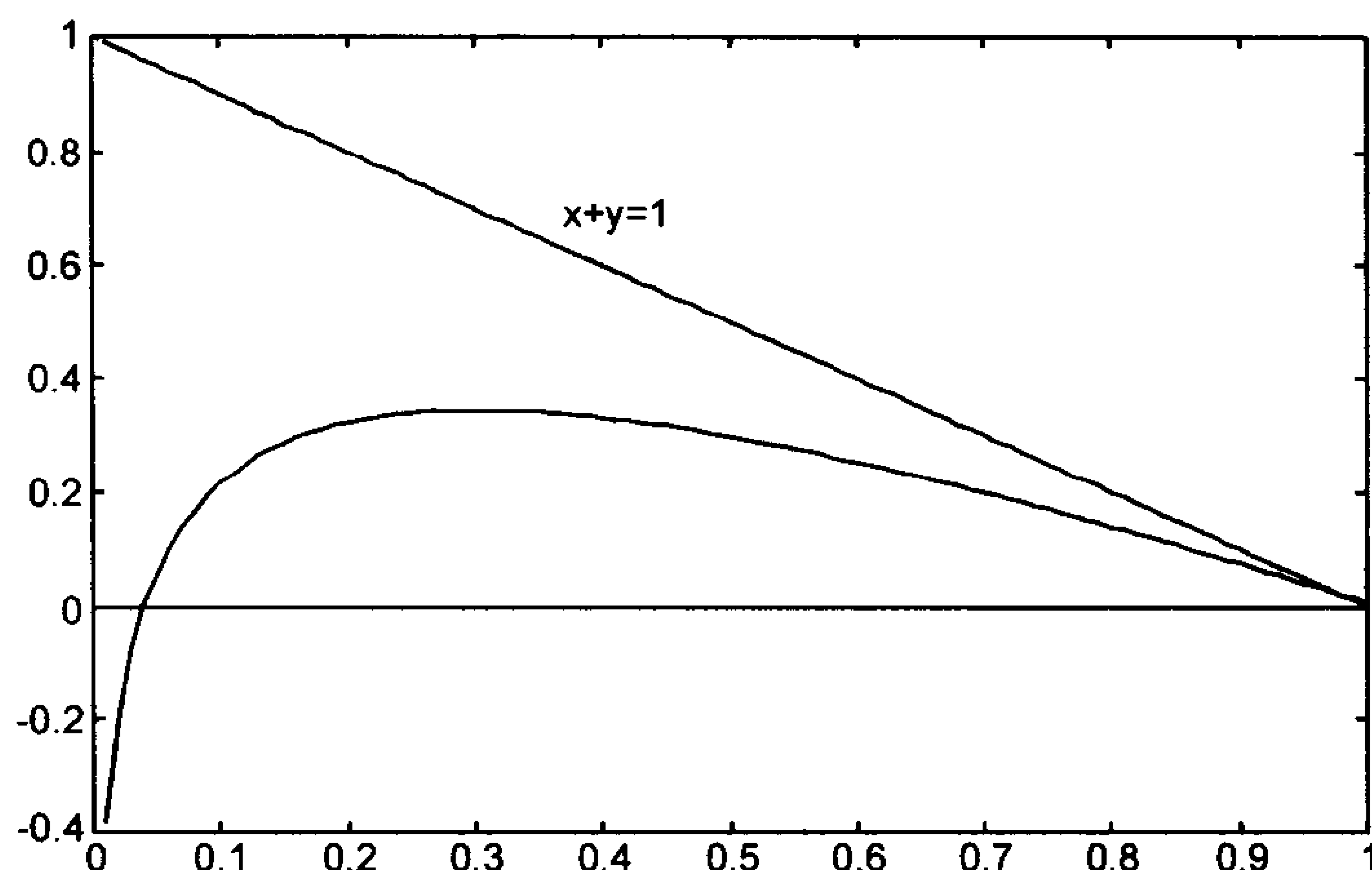


图 2-17 相轨线(2.23)的图形

是微分方程(2.20)、(2.21)的相轨线. 根据 x,y 的含义,函数 $y(x)$ 的定义域为 $x\geqslant 0, y\geqslant 0, x+y\leqslant 1$. 用 MATLAB 不难作出相轨线(2.23)的图形,如图 2-17. 由方程(2.20)知 $x'<0$,所以轨线的方向是从右向左,又由方程(2.22)知,当 $x=1/\sigma$时 $dy/dx=0$,$y(x)$取得极大值.

问题是考察传染过程结束($t\to\infty$)时健康人比例 x_∞,因为 $t\to\infty$ 时病人比例 $y_k=0$(可以从图 2-17 直观地看出),所以根据(2.23)式,x_k(为简单起见以下仍记作 x)是如下方程的根

$$f(x)=\frac{1}{\sigma}\ln\frac{x}{x_0}-x+(x_0+y_0)=0, \tag{2.24}$$

这里用牛顿切线法的 MATLAB 主程序 newtonqx.m 求方程(2.24)的近似根.

先建立名为 fnq.m 和 dfnq.m 的 M 文件

```
function y = fnq(x)
        a =1;b =0.3; sigma =a/b; x0 =0.98;y0 =0.02;
        y =1/sigma * log(x/x0) -x +x0 +y0;
function y =dfnq(x)
          a =1;b =0.3;sigma =a/b; y =1/sigma/x -1;
```

这里迭代的初始值取 0.07,然后在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk,piancha,xdpiancha] =newtonqx(0.07,1e -5, 1e -5,100)
```

运行后输出结果,整理得:迭代次数 $k=5$,方程的近似根 $x_k=0.039\ 9$, $y_k=-5.506\ 9e-013$.

读者不妨取其他初始值,观察会有什么结果.

对于题目给出的 a,b,x_0,y_0 的不同取值,选择合适的迭代初始值,计算结果

如表 2－11 所示. 可以看出,若日接触率 a 由 1 减小到 0.6、日治愈率 b 由 0.3 增加到 0.5,传染过程结束时健康人比例 x_k 将从 0.039 9 提高到 0.196 5,再到 0.624 5,显然这与提高卫生水平和医疗水平有利于制止传染病的蔓延是一致的;若用接种预防疫苗的办法使初始时刻易感染者(健康人)的比例降低为 $x_0=0.70$,则 x_k 将从 0.039 9 提高到 0.084 0 ($a=1, b=0.3$),或者从 0.196 5 提高到 0.305 6 ($a=0.6, b=0.3$),这也与提高免疫率有利于制止传染病的蔓延相一致,但是当 $a=0.6, b=0.5$ 时,x_k 反而从 0.624 5 略降至 0.623 3,似乎有悖常识,读者可以利用图形解释这一现象.

表 2－11 $t\to\infty$ 时健康人比例 x_k 的计算结果

a	b	x_0	y_0	k	x_k	y_k	$\lvert x_k-x_{k-1}\rvert$	$\lvert x_k-x_{k-1}\rvert/\lvert x_k\rvert$
1	0.3	0.98	0.02	5	0.039 9	－5.506 9e－013	7.653 3e－008	1.916 2e－006
0.6	0.3	0.98	0.02	5	0.196 5	－1.105 7e－010	4.131 7e－006	2.103 0e－005
0.6	0.5	0.98	0.02	8	0.624 5	－1.734 7e－017	2.706 2e－009	4.333 4e－009
1	0.3	0.70	0.02	4	0.084 0	－3.504 1e－016	3.889 0e－009	4.628 0e－008
0.6	0.3	0.70	0.02	6	0.305 6	－4.556 3e－011	4.125 6e－006	1.350 0e－005
0.6	0.5	0.70	0.02	8	0.623 3	－1.734 7e－017	2.414 7e－009	3.873 8e－009

2.6.4 求$\sqrt[n]{c}$的方法及其 MATLAB 程序

(一) 求$\sqrt[n]{c}$ (当 n 是偶数时,$c>0$)的公式

设$\sqrt[n]{c}=x$,则 $f(x)=x^n-c=0$. 由牛顿迭代公式(2.12)得

$$x_k=x_{k-1}-\frac{x_{k-1}^n-c}{nx_{k-1}^{n-1}}\ (\text{其中 } k=1,2,3,\cdots), \tag{2.25}$$

称为求 c 的 n 次根$\sqrt[n]{c}$的迭代公式.

(二) 求 c 的 n 次根$\sqrt[n]{c}$的 MATLAB 主程序

用 c 的 n 次根$\sqrt[n]{c}$的迭代公式求$\sqrt[n]{c}$的近似值 x_k(精度为 tol)需要自行编制程序.

求 c 的 n 次根$\sqrt[n]{c}$ (当 n 是偶数时,$c>0$)的 MATLAB 主程序

输入的量:初始值 x_0,根指数 n,被开方数 c,$\sqrt[n]{c}$的误差限 *tol*,迭代次数的最大值 *gxmax*.

输出的量:迭代序列 $\{x_k\}$、迭代 k 次得到的迭代值 x_k(迭代次数达到最大值 *gxmax* 时,运行后输出信息:'请注意:迭代次数超过给定的最大值 gxmax')、相邻两次迭代的偏差 *piancha* $=\lvert x_k-x_{k-1}\rvert$和它的偏差的相对误差 *xdpiancha* $=\lvert x_k-x_{k-1}\rvert/\lvert x_k\rvert$的值.

根据求 c 的 n 次根$\sqrt[n]{c}$的迭代公式(2.25)和已知条件,编写一个名为 kainfang.m 的 M 文件:

```
function [k,xk,yk,piancha,xdpiancha] = kainfang(x0,c,n,tol, gxmax)
     x(1) = x0;
    for i = 1: gxmax
      u(i) = (x(i)^n - c)/(n * x(i)^(n - 1)); x(i + 1) = x(i) - u(i);
      piancha = abs(x(i + 1) - x(i));
      xdpiancha = piancha/( abs(x(i + 1)) + eps);
      i = i + 1; xk = x(i);yk = fnq(x(i));
      [(i - 1),xk,yk,piancha,xdpiancha]
              if (piancha < tol) |(xdpiancha < tol)
                  k = i - 1;xk = x(i); yk = fnq(x(i));[(i - 1),xk,
                  yk,piancha,xdpiancha];
                 return;
              end
    end
    if i > gxmax
          disp('请注意:迭代次数超过给定的最大值 gxmax.')
          k = i - 1;xk = x(i);yk = fnq(x(i));
          [(i - 1),xk,yk,piancha,xdpiancha]
          return;
    end
```

(三) 应用举例

例 2.6.5 求 $\sqrt{113}$,要求精度为 10^{-5}.

解 本题介绍四种解法.

方法 1 用求 c 的 n 次根$\sqrt[n]{c}$ (当 n 是偶数时,$c>0$)的 MATLAB 程序计算.

取初始值 $x_0=10$,根指数 $n=2$,被开方数 $c=113$,近似根的精度 $tol=10^{-5}$,迭代的最大次数 $gxmax=100$. 在工作区间输入程序

```
>> [k,xk,yk,piancha,xdpiancha] = kainfang(10,113,2,1e 5,100)
```

运行后输出结果

```
k = 4,piancha = 1.610800381968147e - 011,
xdpiancha = 1.515313534117706e - 012
xk  = 10.63014581273465,yk  = 1.710910332060509e + 009
```

可见,$\sqrt{113}\approx 10.630\ 15$,满足精度 10^{-5}.

方法 2 用牛顿迭代公式(2.12)计算.

设$\sqrt{x}=113$,则 $x^2-113=0$,记 $f(x)=x^2-113$,$f'(x)=2x$. 由牛顿迭代公式

得，$x_{k+1}=x_k-\frac{x_k^2-113}{2x_k}$，即 $x_{k+1}=\frac{1}{2}\left(x_k+\frac{113}{x_k}\right)$（其中 $k=0,1,2,3,\cdots$）

取初始值 $x_0=10$，计算结果列入表 2－12.

表 2－12

迭代次数 k	偏差 $\lvert x_{k+1}-x_k\rvert$	根的近似值 x_k
1	0.650 000	10.650 000
2	0.019 836	10.630 164
3	0.000 019	10.630 146
4	0.000 000	10.630 146

因为，迭代次数 $k=4$ 时，偏差 $\lvert x_4-x_3\rvert<10^{-8}$，满足精度 10^{-5}，所以，$\sqrt{113}\approx10.630\ 15$.

方法 3 用牛顿切线法的 MATLAB 主程序计算.

分别建立名为 fnq.m 和 dfnq.m 的 M 文件

```
function y = fnq(x)
        y = x^2 - 113;
function y = dfnq(x)
        y = 2 * x;
```

在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk,piancha,xdpiancha] = newtonqx(10, 1e-5, 1e-5,100)
```

运行后，将输出的结果列入下表 2－13. 迭代 $k=4$ 次，得到精度为 10^{-5} 的结果 $\sqrt{113}\approx10.630\ 15$.

表 2－13

k	*piancha*	*xdpiancha*	x_k	y_k
1	0.650 000	0.061 033	10.650 000	0.422 500
2	0.019 836	0.001 866	10.630 164	0.000 393
3	0.000 019	0.000 002	10.630 146	0.000 000
4	0.000 000	0.000 000	10.630 146	0.000 000

方法 4 在 MATLAB 工作空间输入程序

```
>> 113^(1/2)
```

运行后输出

```
ans = 10.63014581273465
```

经过四舍五入后，得到精度为 10^{-5} 的结果 $\sqrt{113}\approx10.630\ 15$.

2.6.5 牛顿切线法的缺陷

(一) 一个实例

我们通过下面的例子来研究牛顿切线法的缺陷.

例 2.6.6 用牛顿切线法求方程 $f(x)=4-e^x\sin x=0$ 的四个最小实根 x_k $(k=1,2,3,4)$,要求 $|f(x_k)|<10^{-5}$. 如何确定初始值? 初始值选取对收敛速度有何影响?

解 (1) 确定初始值.

确定初始值的方法有多种,如作图法、试值法、逐步搜索法等. 这里用作图法确定方程 $4-e^x\sin x=0$ 的四个最小实根的初始值.

在工作窗口输入程序

```
>>x = -4: 0.1: 4; e=exp(1); f=4-e.^x.*sin(x); plot(x,f)
  grid,gtext('f=4-e^xsin(x)')
```

运行后画出函数 $f(x)=4-e^x\sin x$ 的图形(见图 2-18),可见两个最小实根分别在区间 [1,2]和[2,3]内. 为确定方程 $f(x)=0$ 的近似解,再画函数 $f(x)=4-e^x\sin x$ 在区间[1,3]上的图形(见图 2-19),可见两个最小实根的初始值可以分别取为 1.4 和 2.9. 最后画出函数在[2,10]上的图形(见图 2-20),可见另外的两个最小实根的初始值可以分别取为 6.3 和 9.4.

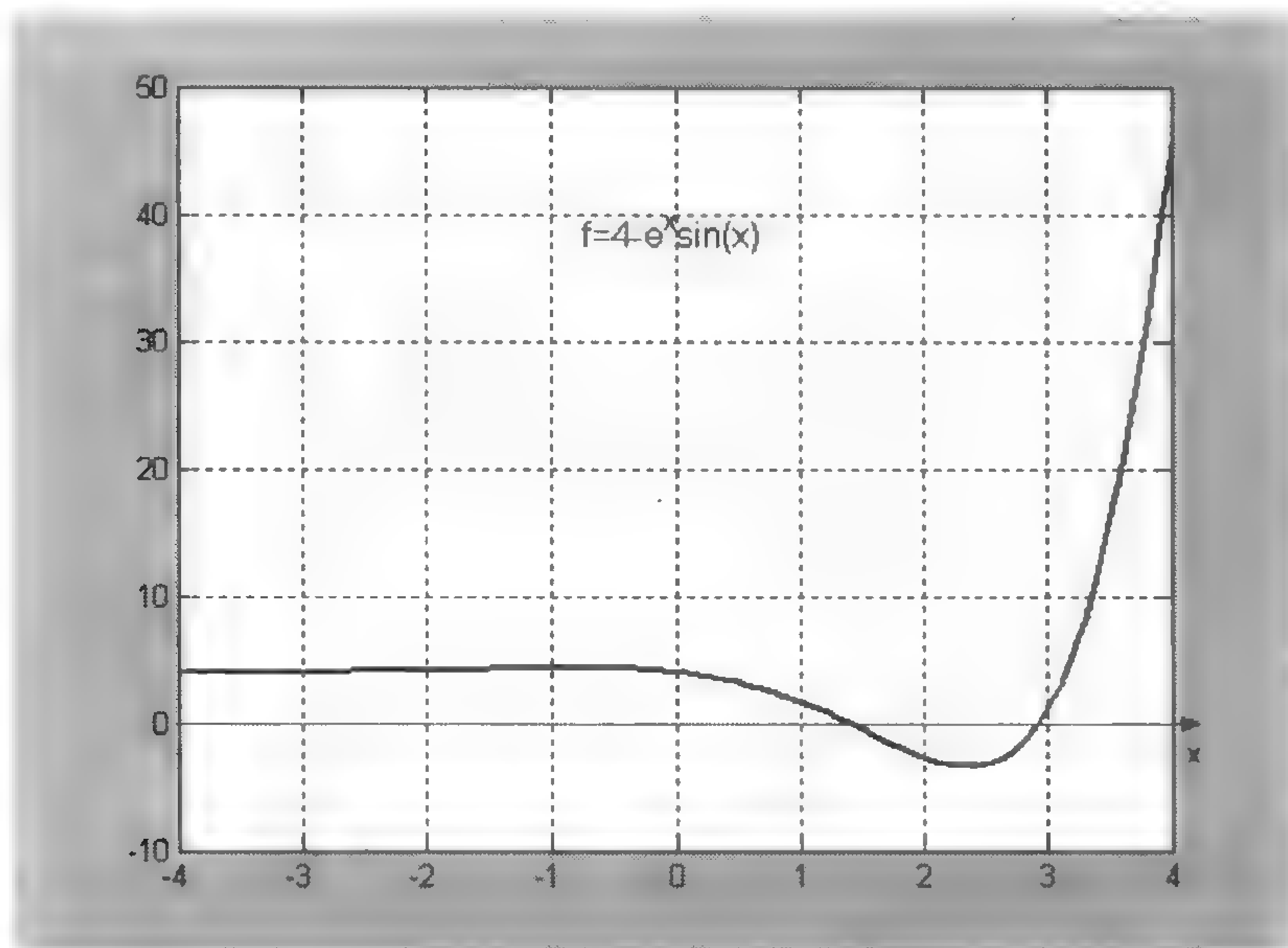


图 2-18

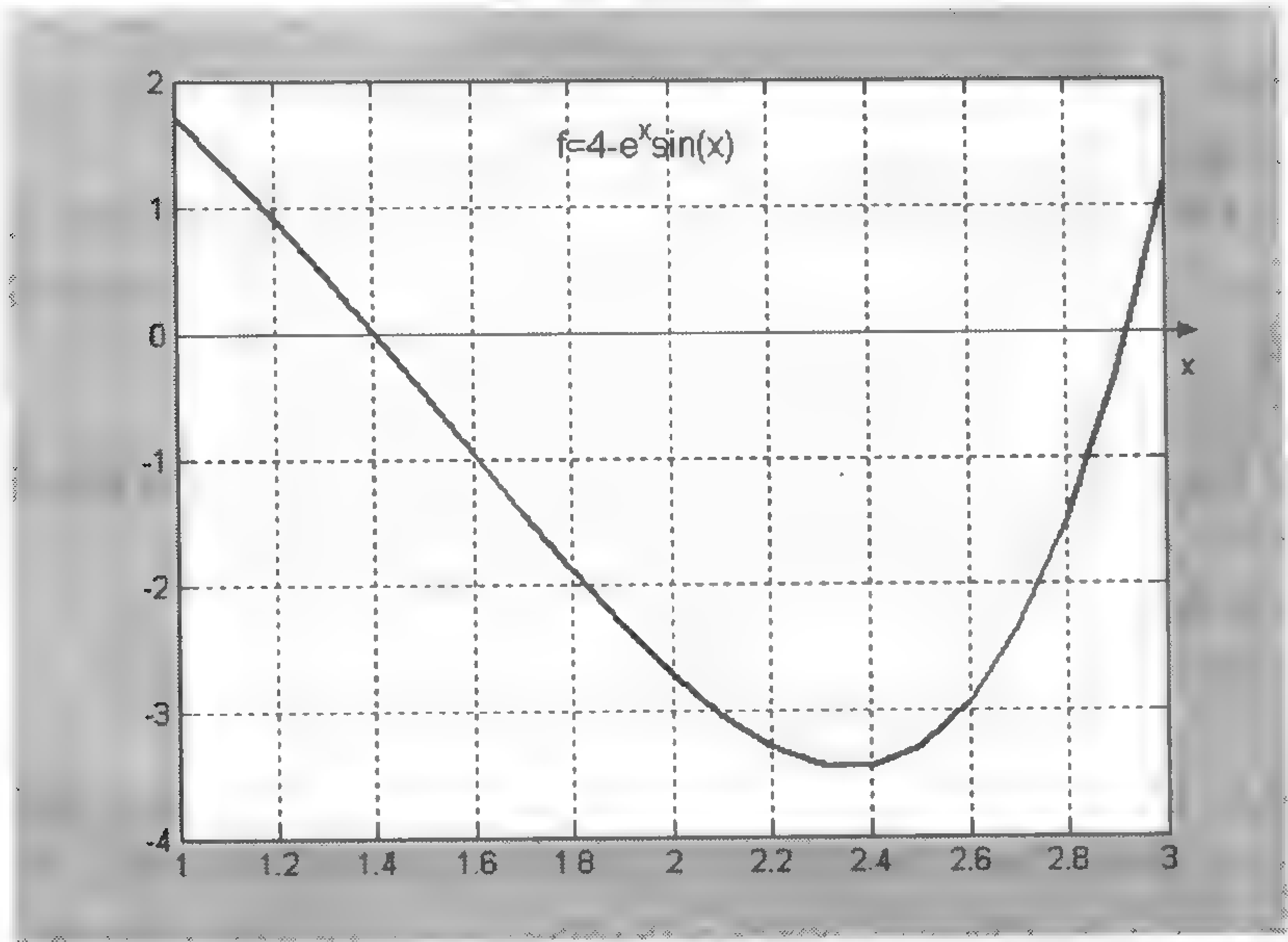

f=4-e^x sin(x)
x
2
1
0
-1
-2
-3
-4
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3

图 2－19

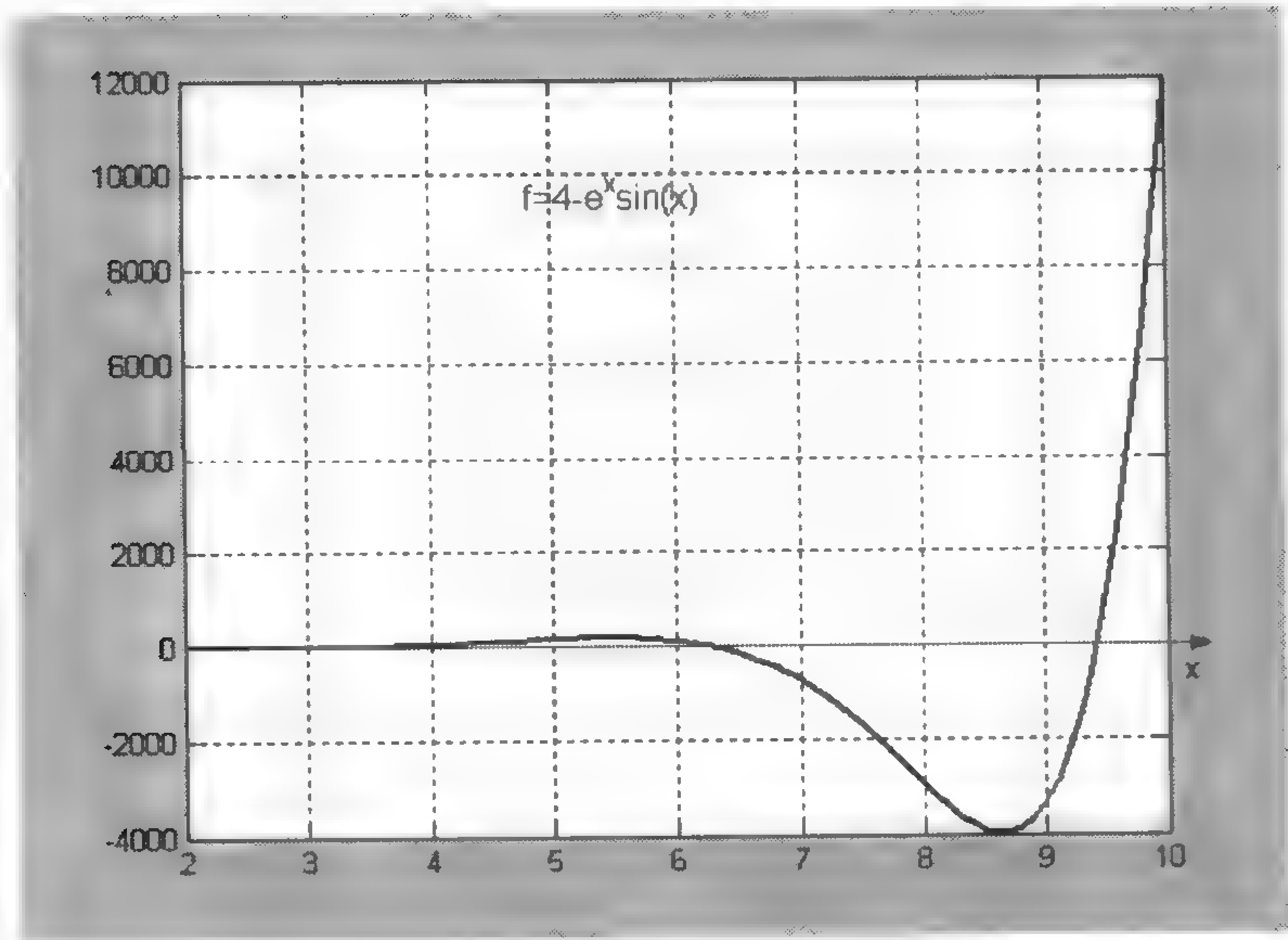

f=4-e^x sin(x)
x
12000
10000
8000
6000
4000
2000
0
-2000
-4000
2
3
4
5
6
7
8
9
10

图 2－20

(2) 用牛顿切线法求方程 $e^x\sin x=4$ 的四个最小正实根，精确到 10^{-5}.

① $f(x)=4-e^x\sin x, f'(x)=-e^x(\sin x+\cos x)$，牛顿迭代公式

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)} \quad (k=0,1,2,\cdots).$$

② 建立名为 fnq.m 的 M 文件

```
function y = fnq(x)
 e = exp(1); y = 4 - e^ x * sin(x);
```

③ 建立名为 dfnq.m 的 M 文件

```
function y = dfnq(x)
  e = exp(1);y = - e^ x * (sin(x) + cos(x));
```

④ 在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk,piancha,xdpiancha] = newtonqx(1.4,1e - 5, 1e - 5,100)
```

⑤ 运行后输出结果

```
k =
   2
xk =
1.40081184023081
yk  =
 -6.217248937900877e - 015
piancha =
         9.682842816438608e - 008
xdpiancha =
          6.912307947684956e - 008
```

经整理填入表 2 - 14. 经过迭代 $k=2$ 次，求得 $|f(x_k)|<10^{-5}$的最小实根为 $x_1\approx1.400\ 81$.

表 2 - 14

迭代次数 k	x_k	y_k	*piancha*	*xdpiancha*
1	1.400 812	-0.000 001	0.000 812	0.000 580
2	1.400 812	-0.000 000	0.000 000	0.000 000

在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk,piancha,xdpiancha] = newtonqx(2.9,1e - 5, 1e - 5,100)
```

运行后输出结果经整理填入表 2 - 15. 经过迭代 $k=3$ 次，求得精确到 10^{-5}的第二小实根为 $x_2\approx2.925\ 32$.

表 2－15

迭代次数 k	x_k	y_k	*piancha*	*xdpiancha*
1	2.926 180	0.012 227	0.026 180	0.008 947
2	2.925 322	0.000 013	0.000 859	0.000 294
3	2.925 321	0.000 000	0.000 001	0.000 000

同理可得,所求的另外的两个近似根分别为 6.290 60 和 9.424 46.

(3) 讨论初始值选取对收敛速度有何影响

我们以方程 $e^x \sin x = 4$ 的最小实根 $x_1 \approx 1.400\ 81$(精确到 6 位有效数字)为例,用牛顿切线法讨论初始值选取对收敛速度有何影响.

利用 MATLAB 命令

```
>>[k,xk,yk,piancha,xdpiancha] = newtonqx(x0,1e-7, 1e-7,100)
```

① 取初始值 $x_0 = 1.4$ 时,$|x_1 - x_0| \approx 0.000\ 81$,经过迭代 2 次,求得精确到 6 位有效数字的最小实根为 $x_1 \approx 1.400\ 81$.

② 取初始值 $x_0 = 1$ 时,$|x_1 - x_0| \approx 0.400\ 81$,经过迭代 $k = 4$ 次,求得同样结果.

③ 取初始值 $x_0 = 2$ 时,$|x_1 - x_0| \approx 0.599\ 19$,经过迭代 $k = 5$ 次,求得同样结果.

④ 取初始值 $x_0 = 3$ 时,$|x_1 - x_0| \approx 1.599\ 19$,经过迭代 $k = 4$ 次,求得 $x_2 = 2.925\ 32$是另外一个根. 这是因为$|x_2 - x_0| \approx 0.074\ 68$,即初始值 $x_0 = 3$ 在 x_2 附近,而离 $x_1 \approx 1.400\ 81$ 较远.

⑤ 取初始值 $x_0 = 5.5$ 时,虽然 5.5 在此方程的另一个根 $x_3 = 6.290\ 60$ 附近,$|x_3 - x_0| \approx 0.790\ 60$,但是在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk,piancha,xdpiancha] = newtonqx(5.5,1e-7, 1e-7, 1000)
```

则运行后输出结果

```
请注意:迭代次数超过给定的最大值 gxmax.
```

计算情况如表 2－16 所示.

表 2－16

k	*piancha*	*xdpiancha*	x_k	y_k
1	8535/37	713/730	17477/74	1.960 247e + 102
2	386/1007	22/13533	18156/77	4.352 170e + 101
3	317/2133	26/41225	20501/87	5.278 756e + 100

续表

k	*piancha*	*xdpiancha*	x_k	y_k
4	205/8674	5/49848	11781/50	1.208 723e+099
5	65/114632	1/415531	26625/113	6.848 663e+095
6	*	*	26625/113	2.149 462e+089
7	*	*	26625/113	-2.712 145e+088
8	0	0	26625/113	-2.712 145e+088
9	0	0	26625/113	-2.712 145e+088
10	0	0	26625/113	-2.712 145e+088
⋮	⋮	⋮	⋮	⋮
999	0	0	26625/113	-2.712 145e+088
1000	0	0	26625/113	-2.712 145e+088

由此可见，迭代次数从 $k=8$ 到 1 000 时的迭代值 $x_k=26625/113$，相邻两次迭代的偏差 *piancha* $=|x_k-x_{k-1}|$ 和它的偏差的相对误差 *xdpiancha* $=|x_k-x_{k-1}|/|x_k|$ 的值近似于 0，迭代函数值 $y_k=-2.712\,15e+088$，此迭代序列收敛到 26625/113. 由程序

```
>> x = 26625/113, e = exp(1);
f = 4 - e^(x) * sin(x)
```

运行后的结果

```
x =
2.356194690265487e+002
f =
4.260141448977054e+097
```

可知，$x_k=26625/113\approx235.619\,469\,026\,548\,7$ 不是此方程的根. 由程序

```
>> x = 26625/113; e = exp(1);
   df = -e^x*(sin(x) + cos(x))
```

运行后得

```
df =
2.129334624530098e+102
```

迭代值 $x_k=26625/113$ 也不是函数 $f(x)=4-e^x\sin x$ 的极值点. 但是从图 2-22 可以看出迭代值 $x_k=26625/113$ 与方程 $f(x)=0$ 的根很接近，如果要求的近似根的精度不高时，26625/113 可以作为近似根。从理论上看，因为一阶导数值 $|f'(5.5)|\approx|-0.765\,751|<1$，一般情况下，由初始值 $x_0=5.5$ 产生的迭代序列 $\{x_k\}$ 收敛到此方程的根。实际上，计算机输出的迭代序列 $\{x_k\}$ 不精确地收敛到此方程根，其原因是**计算机硬件只支持有限位机器数的运算**，所以 x_k 与 x_{k-1}

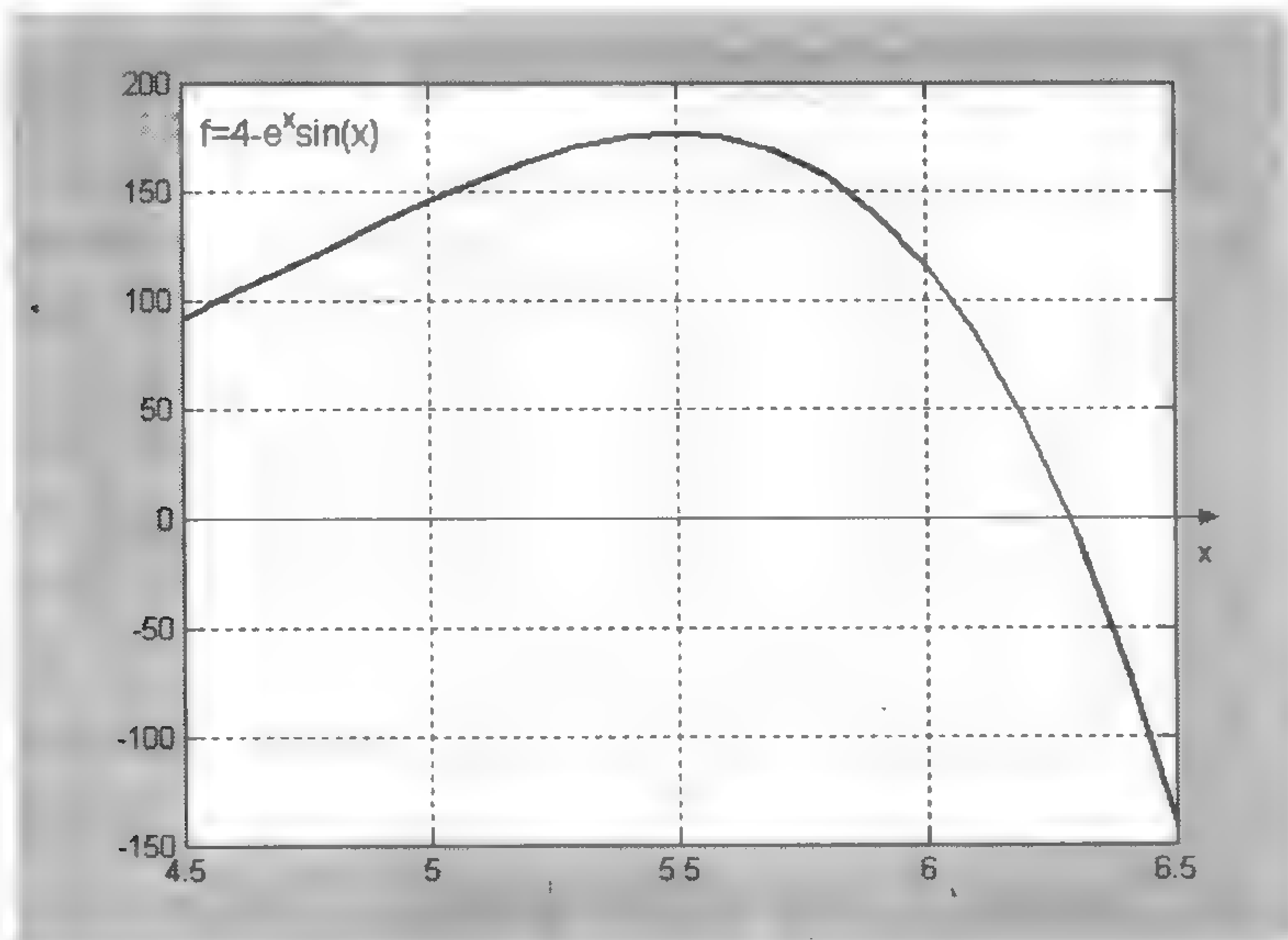


图 2－21

相减时,在计算中引入和传播舍入误差. 因为有效数字的严重损失,导致输出 $|x_k - x_{k-1}|$ 的结果为 0,计算机不能再继续进行真实地计算,所以,最后输出的迭代序列 $\{x_k\}$ 不能精确地收敛到此方程的根. 如果输入程序

```
>> [k,xk,yk,piancha,xdpiancha] =newtonqx(26625/113,1e -7, 1e -7,100)
```

运行后屏幕显示的由初始值 $x_0 = 26625/113$ 产生迭代序列 $\{x_k\}$ 也收敛到 26625/113. 由初始值 $x_0 = 5.5$ 产生的迭代序列 $\{x_k\}$ 不收敛到此方程最近的根的原因是初始值 $x_0 = 5.5$ 与此方程的最近的根 $x_3 = 6.290\ 60$ 的距离 $|x_3 - x_0| \approx 0.790\ 600$,而 $x_0 = 5.5$ 与函数的一个极大值点 $x \approx 5.497\ 79$ 之间的距离是 0.002 212 88 ($f'(5.497\ 787) \approx 4.964\ 53e-005$)(见图 2－21). 由此可见,当初始值 x_0 与方程最近的根 x^* 之间存在极值点时,即使 $|f'(x_0)| < 1$,由初始值 x_0 产生的迭代序列 $\{x_k\}$ 也不一定收敛到此方程最近的根 x^*.

如果我们在靠近此函数的极小值点取初始值 $x_0 = 8.6$ 时(见图 2－20),会得到类似的结果.

⑥ 取初始值 $x_0 = 10$ 时,因为 10 在此方程的另一个根 $x_4 = 9.424\ 46$ 附近,$|x_4 - x_0| \approx 0.575\ 545$,经过迭代 $k = 5$ 次,求得 $x_4 = 9.424\ 46$ 是另外一个根(见图 2－20).

⑦ 取初始值 $x_0 = -1$,最大迭代次数 $gxmax = 100$ 时,输入程序

```
>> [k,xk,yk,piancha,xdpiancha] =newtonqx( -1,1e -7, 1e -7,100)
```

运行后屏幕显示

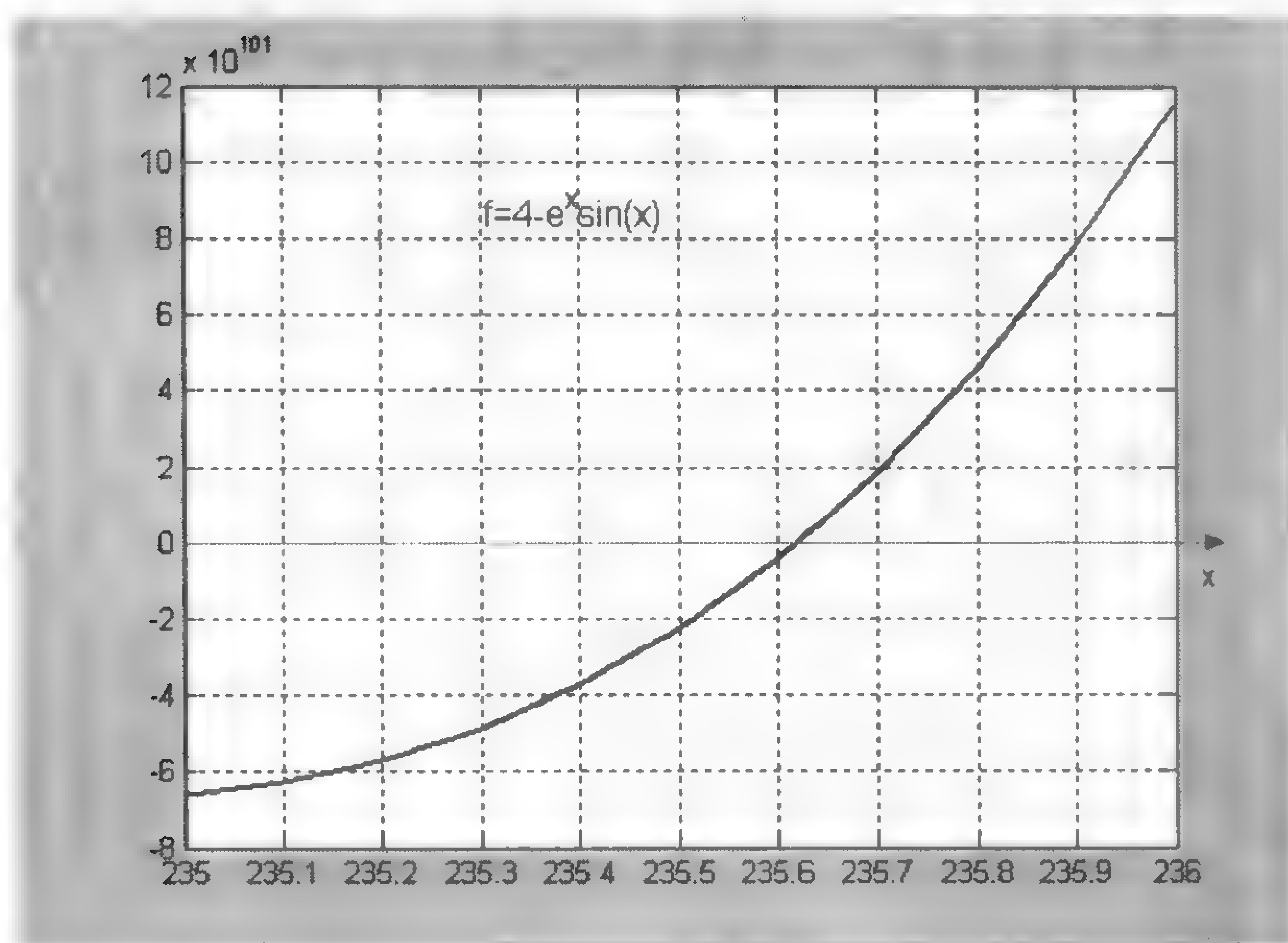


图 2-22

```
请注意:迭代次数超过给定的最大值 gxmax.
k =100
xk = -1.782907298075927e+018
yk =4
piancha =1.801439850948198e+016
xdpiancha =0.01010394568967
```

如果将最大迭代次数改为 gxmax =100000,则运行后屏幕显示

```
请注意:迭代次数超过给定的最大值 gxmax.
k =100000
xk = -1.801421318395326e+021
yk =4
piancha =1.801439850948198e+016
xdpiancha =1.000010287739289e-005
```

因为牛顿切线法是局部收敛的,而 -1 离最近的根 $x_1 \approx 1.40081$ 的 $|x_1 - x_0| \approx 2.40081$,由图 2-18 可见,曲线 $f(x)$ 在初始值 $x_0 = -1$ 附近非常平缓,且 $\lim\limits_{x \to -\infty}(4 - e^x \sin x) = 4$,所以,此时产生的迭代序列 $\{x_k\}$ 发散.

从以上① 至⑦的讨论可见,**用牛顿切线法求方程的根的近似值时,初始值的选取对收敛速度有很大的影响**. 为了研究一般情况,我们取初始值 $x_0 = -1$ 至 10 ,运行后得到的结果列表 2-17.

表 2-17 牛顿切线法求方程 $e^x \sin x=4$ 的近似实根

x_0	x^*	x_k	k	$\lvert\varphi'(x_0)\rvert$	$\lvert x^*-x_0\rvert$	*piancha*	*xdpiancha*	y_k
-1	1.400 81	-1.782 91e+021	>100	139.564	2.400 8	1.801 44e+16	0.010 103 95	4
0	1.400 81	2.925 32	[redacted]	8.000 0	1.400 8	1.064 5e-012	3.639e-013	-2.664 54e-015
1	1.400 81	1.400 81	4	0.356 6	0.400 8	1.871 1e-008	1.336e-008	4.440 89e-016
2	1.400 81	1.400 81	5	1.259 3	0.599 2	1.531 0e-012	1.093e-012	-8.881 78e-016
2.3	2.925 32	NaN	>100	72.799 3	0.625 3	NaN	NaN	NaN
2.356	2.925 32	-1.783 43e+018	>100	6 131 189	0.569 3	1.801 4e+016	0.010 101 01	4
3	2.925 32	2.925 32	4	0.159 4	0.074 7	3.285 0e-009	1.123e-009	-2.664 54e-015
4	2.925 32	2.925 32	7	0.545 5	1.074 7	1.064 5e-012	3.639e-013	-2.664 54e-015
5	6.290 60	2.925 32	7	1.226 6	1.290 6	1.183 6e-010	4.046e-011	-2.664 54e-015
5.4	6.290 60	9.424 46	8	52.658 5	0.890 6	6.901 2e-012	7.323e-013	-5.360 16e-012
5.5	6.290 60	235.619	>100	104 470	0.790 6	0	0	-2.712 15e+088
5.581 2	6.290 60	9.424 46	7	72.687 6	0.709 4	1.564 5e-007	1.660e-008	3.026 09e-010
6	6.290 60	6.290 60	5	1.198 9	0.290 6	5.011 6e-008	7.967e-009	-1.442 40e-012
7	6.290 60	6.290 60	6	0.494 9	0.709 4	1.293 5e-011	2.056e-012	5.773 16e-015
8	9.424 46	6.290 60	6	0.403 8	1.424 5	2.092 9e-007	3.327e-008	-2.364 69e-011
8.6	9.424 46	449.248	>100	321.265	0.824 5	0	0	-8.745 45e+180
9	9.424 46	9.424 46	6	3.012 2	0.424 5	7.391 5e-009	7.843e-010	-5.360 16e-012
10	9.424 46	9.424 46	5	0.477 4	0.575 5	4.665 8e-007	4.951e-008	2.700 37e-009

由表 2-17 和本例的图形可见,由牛顿迭代公式产生的迭代序列$\{x_k\}$收敛速度与选取的初始值x_0与根x^*距离和点$(x_0,f(x_0))$附近的曲线的陡峭程度及其$|\varphi'(x_0)|=\left|\frac{f(x_0)f''(x_0)}{[f'(x_0)]^2}\right|$的大小等因素有关. 本例中,取初始值$x_0$与根$x^*$距离$|x^*-x_0|<1.08$且$|\varphi'(x_0)|<1$时,产生的迭代序列$\{x_k\}$收敛到与$x_0$最近的根$x^*$,并且$|x^*-x_0|$和$|\varphi'(x_0)|$值越小,迭代序列$\{x_k\}$收敛速度越快(见$x_0=1,3,4,7,10$),这与牛顿切线法具有局部收敛性一致;如果$|x^*-x_0|<0.6$且$1<|\varphi'(x_0)|<3.02$时,有些产生的迭代序列$\{x_k\}$也收敛到与$x_0$最近的根$x^*$(如$x_0=2,6,9$等);在函数极值点的附近取初始值$x_0$,产生的迭代序列$\{x_k\}$有的收敛,但是收敛的值不是方程的根(如$x_0=5.5$, 8.6 等);如果$|x^*-x_0|<1.5$或$|\varphi'(x_0)|<73$时,产生的迭代序列$\{x_k\}$有的收敛到根$x^*$,但是$x^*$不一定是距离方程最近的根(如$x_0=0$, 5 ,5.4,5.581 2, 8 等);如果$|\varphi'(x_0)|>100$,且$|x^*-x_0|>2$时,产生的迭代序列$\{x_k\}$发散(如$x_0=-1$等).

(二)牛顿切线法的缺陷

一般来说,用牛顿切线法求方程$f(x)=0$的根,如果取的初始值x_0与方程$f(x)=0$最近的根x^*的距离$|x_0-x^*|<1$,且$|\varphi'(x_0)|=\left|\frac{f(x_0)f''(x_0)}{[f'(x_0)]^2}\right|<1$,当$x_0$与$x^*$之间不存在极值点时,则由式(2.12)生成的迭代序列$\{x_k\}$收敛(或近似收敛)到方程$f(x)=0$的根x^*;当x_0与x^*之间存在极值点时,则由式(2.12)生成的迭代序列$\{x_k\}$收敛(或近似收敛)到方程$f(x)=0$的根x,但是x不一定是与初始值x_0距离最近的根. 如果$|x_0-x^*|>1$或$|\varphi'(x_0)|>1$,则可能会出现一些问题,我们讨论如下:

(1) 取初始值x_0满足$|x_0-x^*|>1$且$|\varphi'(x_0)|<1$,由式(2.12)生成的迭代序列$\{x_k\}$不一定收敛到要求的方程$f(x)=0$根. 比如例 2.6.6 中,初始值$x_0=8$,$|x_0-x^*|=1.424\,5>1$且$|\varphi'(x_0)|=0.403\,8<1$,但是,由式(2.12)生成的迭代序列$\{x_k\}$没有收敛到距离 8 最近的根$x^*\approx9.424\,5$,而是收敛到另一个根 6.290 6.

(2) 取初始值x_0,使$\varphi'(x_0)=0$且$|x_0-x^*|\neq0$,则由式(2.12)生成的迭代序列$\{x_k\}$不一定收敛,有时发散,有时其中的项重复或基本重复.

例如,方程$8xe^{-x}=0$,取初始值$x_0=2$时,$\varphi'(2)=0$,则由式(2.12)生成的迭代序列$\{x_k\}$为$x_1=4.000\,0$, $x_2=5.333\,3$, $x_3=6.564\,1$, …, $x_{10\,000}=53.136\,7$,…缓慢地发散到$+\infty$.

再如,方程$f(x)=x^3-x-3=0$有一个实根$x^*\approx1.671\,70$,取初始值$x_0=0$时,$|x_0-x^*|=1.671\,7>1$且$|\varphi'(x_0)|=0<1$,但是,由式(2.12)生成的迭代序

列$\{x_k\}$基本重复(类似于周期性的变化),不收敛,见表 2-18.

表 2-18 牛顿切线法求方程$f(x)=x^3-x-3=0$的根

k	*piancha*	*xdpiancha*	x_k	y_k
1	3.000 0	1.000 0	-3.000 0	-27.000 0
2	1.038 5	0.529 4	-1.961 5	-8.585 7
3	0.814 4	0.709 9	-1.147 2	-3.362 5
4	1.140 6	173.359 5	-0.006 6	-2.993 4
5	2.993 8	0.997 8	-3.000 4	-27.010 1
6	1.038 6	0.529 4	-1.961 8	-8.588 7
7	0.814 4	0.709 7	-1.147 4	-3.363 3
8	1.140 2	157.130 0	-0.007 3	-2.992 7
9	2.993 2	0.997 6	-3.000 5	-27.012 3
10	1.038 6	0.529 4	-1.961 9	-8.589 3
11	0.814 4	0.709 7	-1.147 5	-3.363 4
12	1.140 1	154.013 2	-0.007 4	-2.992 6
13	2.993 1	0.997 5	-3.000 5	-27.012 8
⋮	⋮	⋮	⋮	⋮
995	0.814 4	0.709 7	-1.147 5	-3.363 5
996	1.140 1	153.108 2	-0.007 4	-2.992 6
997	2.993 1	0.997 5	-3.000 5	-27.013 0
998	1.038 6	0.529 4	-1.961 9	-8.589 5
999	0.814 4	0.709 7	-1.147 5	-3.363 5
1 000	1.140 1	153.108 2	-0.007 4	-2.992 6
⋮	⋮	⋮	⋮	⋮

(3) 如果$f(x)=0$中的函数$f(x)$在区间$[a,+\infty)$单调递减且$f(x)>0$,取初始值$x_0>a$,则生成的迭代序列$\{x_k\}$可能发散到$+\infty$.

例如,方程$f(x)=4x\mathrm{e}^{-3x}=0$,取初始值$x_0=2/3$时,计算结果见表 2-19. 生成的迭代序列$\{x_k\}$缓慢地发散到$+\infty$,对应的函数序列$\{f(x_k)\}$收敛到0. 即使在牛顿切线法的 MATLAB 主程序中设计终止评定条件$\frac{2|x_{k+1}-x_k|}{|x_k|+10^{-6}}$,迭代 296 次,得到的迭代值$x_{296}=20.454\ 6$,相邻两个迭代值的偏差为 *piancha* = 8.248 9e -010,其绝对误差为 *xdpiancha* = 4.032 8e -011,迭代值的函数值为$y_k=1.831\ 6\mathrm{e}-025$,有可能错误地将$x=296$作为根. 由于这个原因,需要画图. 输入程序

```
>> x = 0: 0.01: 3;e = exp(1);f = 4 * x. * exp( -3 * x);
plot(x,f),grid,gtext('f =4x exp( -3x)')
```

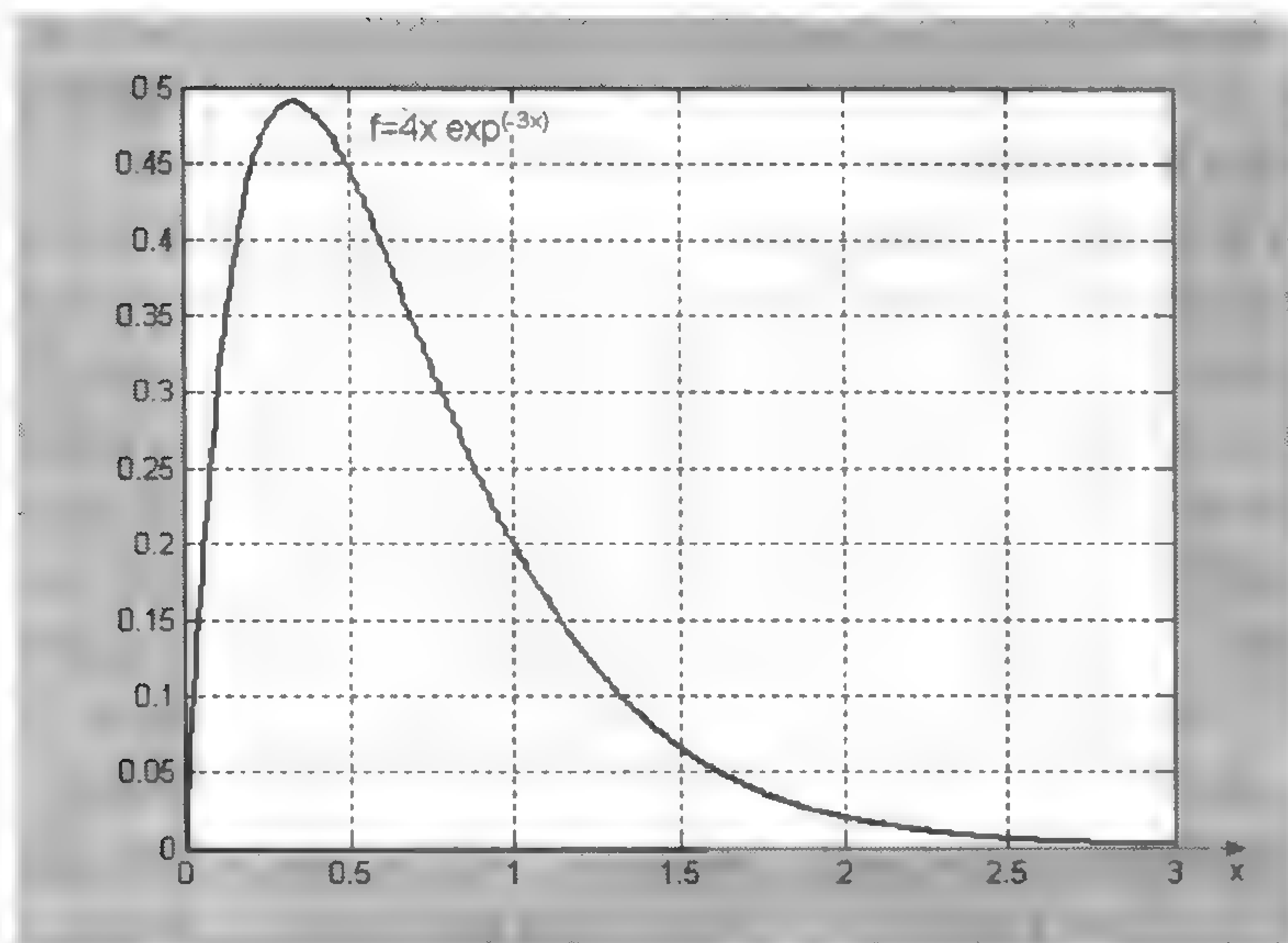


图 2-23

由运行后输出的图 2-23 可以看出,函数 $f(x)$ 在区间 $[2/3, +\infty)$ 内是单调递减的且 $f(x)>0$.

表 2-19 牛顿切线法求 $4xe^{-3x}=0$ 根

k	*piancha*	*xdpiancha*	x_k	y_k
1	0.666 7	0.500 0	1.333 3	0.097 7
2	0.444 4	0.250 0	1.777 8	0.034 3
3	0.410 3	0.187 5	2.188 0	0.012 3
4	0.393 2	0.152 3	2.581 3	0.004 5
5	0.382 8	0.129 1	2.964 0	0.001 6
6	0.375 6	0.112 5	3.339 6	0.000 6
7	0.370 3	0.099 8	3.709 9	0.000 2
8	0.366 2	0.089 8	4.076 1	0.000 1
9	0.363 0	0.081 8	4.439 2	0.000 0
10	0.360 4	0.075 1	4.799 6	0.000 0
⋮	⋮	⋮	⋮	⋮
98	0.037 5	0.002 6	14.459 6	0.000 0
99	0.042 4	0.002 9	14.417 1	0.000 0
100	0.048 8	0.003 4	14.368 3	0.000 0
⋮	⋮	⋮	⋮	⋮

(4) 如果取的初始值 x_0 不满足牛顿切线法局部收敛的条件,即 $|\varphi'(x_0)|\geq 1$,那么,由式(2.12)生成的迭代序列 $\{x_k\}$ 不一定收敛,此时,牛顿切线法求方程 $f(x)=0$ 的根有如下缺陷:

① 当靠近函数 $f(x)$ 的极值点取初始值 x_0 时,由式(2.12)生成的迭代序列 $\{x_k\}$ 的敛散性变化很大. 如例 2.6.6 中⑤,在极大值点 $x\approx5.4978$ 附近取初始值 $x_0=5.5$ 时,由式(2.12)生成的迭代序列 $\{x_k\}$ 收敛到 $x_k=235.619$,但 x_k 不是方程的根;在此极大值点附近取初始值 $x_0=5.4$ 时,迭代序列 $\{x_k\}$ 收敛到方程的根 9.4245,而不是收敛到离它最近的根 6.2906. 在极小值点 $x\approx8.6394$ 附近取不同的初始值 $x_0=8.6$ 和 8 会得到类似的结果. 但是在极小值点 $x\approx2.3562$ 附近取初始值 $x_0=2$ 时,由式(2.12)生成的迭代序列 $\{x_k\}$ 收敛到方程的离它最近的根 $x^*=1.4008$;取 $x_0=2.3$ 时,会出现被零除的错误,生成的序列 $\{x_k\}$ 发散;取 $x_0=2.356$ 时,生成的序列 $\{x_k\}$ 向 $-\infty$ 发散,请读者自己动手做做看.

② 当初始值 x_0 与要求的根 x^* 距离较远,且点 $(x_0,f(x_0))$ 附近的曲线比较平缓,即 $|f'(x_0)|\ll 1$ 时,生成的序列 $\{x_k\}$ 可能发散. 如例 2.6.6 的⑦取初始值 $x_0=-1$,$f'(-1)=0.1108$ 的情形.

③ 当初始值 x_0 与要求的根 x^* 距离较远,由式(2.12)生成的迭代序列 $\{x_k\}$ 不一定收敛到离它最近的根. 如例 2.6.6 的表 2-17 中,离初始值 $x_0=0$ 最近的根是 $x^*=1.4008$,$|x^*-x_0|=1.4008$,且 $f'(0)=-1$. 但是,由式(2.12)生成的迭代序列 $\{x_k\}$ 收敛到离它较远的根 $x^*=2.9253$ 上,这是因为牛顿切线法是局部收敛的.

④ 当初始值 x_0 与要求的根 x^* 距离小于 1 时,由式(2.12)生成的迭代序列 $\{x_k\}$ 不一定收敛到离它最近的根. 例如 $x_0=5.5812$ 和 7 到它们最近的根 6.2906的距离都是 0.7094,但是前者生成的迭代序列 $\{x_k\}$ 收敛到离它较远的根 $x^*=9.4245$,而后者收敛到离它最近的根 $x^*=6.2906$,这是因为 $|\phi'(5.5812)|=72.6876$ 比 1 大得多,所以 $\{x_k\}$ 不是局部收敛的;而 $|\phi'(7)|=0.4949<1$,所以 $\{x_k\}$ 是局部收敛的. 再如 $x_0=2$ 到它们最近的根 1.4008 的距离是0.5992,它比 1 小,且 $|\phi'(2)|=1.2593>1$,但是它生成的迭代序列 $\{x_k\}$ 收敛到离它最近的根 $x^*=1.4008$,所以 $\{x_k\}$ 是局部收敛的.

⑤ 当初始值 x_0 使 $|x^*-x_0|\geq0.7$ 且 $|\phi'(x_0)|>1$,则生成的迭代序列 $\{x_k\}$ 有可能离散振荡. 例如,$f(x)=5\arctan 2x$,取出值 $x_0=0.7$,则 $|\phi'(0.7)|=2.6615>1$,$|x^*-x_0|=0.7$.

在 MATLAB 工作区输入程序

```
>> [k,xk,yk,piancha,xdpiancha] = newtonqx(0.7,1e-7, 1e-7, 9000)
```

运行后输出结果

```
ans =    1.0000   -0.7068   -4.7756    1.4068    1.9904
ans =    2.0000    0.7251    4.8354    1.4319    1.9748
```

```
ans =    3.0000   -0.7753   -4.9901    1.5004    1.9352
ans =    4.0000    0.9235    5.3729    1.6988    1.8395
ans =    5.0000   -1.4468   -6.1903    2.3703    1.6383
ans =    6.0000    4.3552    7.2825    5.8019    1.3322
ans =    7.0000  -51.6249   -7.8056   55.9800    1.0844
ans =  1.0e+003*
        0.0080    8.2703    0.0079    8.3219    0.0010
ans =  1.0e+008*
        0.0000   -2.1486   -0.0000    2.1487    0.0000
ans =  1.0e+016*
        0.0000    2.8436    0.0000    2.8436    0.0000
ans =  1.0e+016*
        0.0000   -0.6935   -0.0000    3.5371    0.0000
ans =  1.0e+016*
        0.0000    2.8436    0.0000    3.5371    0.0000
ans =  1.0e+016*
        0.0000   -0.6935   -0.0000    3.5371    0.0000
ans =  1.0e+016*
        0.0000    2.8436    0.0000    3.5371    0.0000
ans =  1.0e+016*
        0.0000   -0.6935   -0.0000    3.5371    0.0000
     ......................................................................
ans =  1.0e+016*
        0.0000    2.8436    0.0000    3.5371    0.0000
请注意:迭代次数超过给定的最大值gxmax.
     k =9000,piancha =3.5371e+016,xdpiancha =1.2439,
xk =2.8436e+016,yk =7.8540
```

可见,迭代第 $k=11$ 次以后,相邻两次迭代的偏差 *piancha* = 3.537 1e+016,迭代值 x_k 在 −0.693 5e+016 和 2.843 6e+016 之间振荡.

上述情况表明,为了按真实的情况说明结果,有时只根据输出的迭代序列 $\{x_k\}$ 的值 x_k 判断是不够的,因为即使 $\{x_k\}$ 收敛,收敛的值也不一定是方程的根. 所以,还需要根据输出的迭代值的函数值 y_k,相邻两次迭代的偏差及其相对误差 *xdpiancha* 的值综合起来分析迭代序列 $\{x_k\}$ 是否真正收敛到要求的根. 另外,当求根算法在给定的最大迭代值范围内没有找到根时,发出警告来提醒用户是十分必要的(见牛顿切线法的 MATLAB 主程序). 为了克服以上缺陷,我们可以采取以下方法取初始值 x_0,求出方程 $f(x)=0$ 的根 x^* 的近似值 x_k.

步骤1 首先用作图法画出函数$f(x)$的图形,根据图形确定$f(x)=0$的根x^*的位置,并取初始值x_0,满足x_0与x^*的距离$|x_0-x^*|<1$;

步骤2 用牛顿切线法的局部收敛性的MATLAB主程序判别由初始值x_0和式(2.12)产生的迭代序列$\{x_k\}$的敛散性,如果x_0满足

$$0<|\varphi'(x_0)|=\left|\frac{f(x_0)f''(x_0)}{[f'(x_0)]^2}\right|<1,$$

那么,一般情况下,此迭代序列$\{x_k\}$收敛到方程$f(x)=0$的根x^*;

步骤3 用牛顿切线法求方程$f(x)=0$的根x^*的近似值x_k,使其达到要求的精度.

2.6.6 牛顿切线法的加速及其两种MATLAB程序

当x^*是方程$f(x)=0$的m $(m\geqslant 2)$重根时,由牛顿切线公式(2.12)产生的迭代序列$\{x_k\}$收敛到x^*的速度较慢(例如,当$m=2$时,$\{x_k\}$是线性收敛的).我们要提高收敛速度,可以有选择地采用已知根的重数和未知根的重数两种求重根的修正牛顿切线公式.

(一) 已知方程根的重数 m

已知方程$f(x)=0$的根x^*的重数m,求重根x^*的修正牛顿切线公式

$$x_{k+1}=x_k-m\frac{f(x_k)}{f'(x_k)}\ (k=0,1,2,\cdots) \tag{2.26}$$

将产生二阶收敛(平方收敛)的迭代序列$\{x_k\}$.

用迭代公式(2.26)求方程$f(x)=0$的近似根x_k的一般步骤如下:

步骤1 选取初始值x_0和迭代的最大次数$gxmax$以及计算精度ε,计算$f(x_0)$;

步骤2 用迭代公式(2.26)计算x_1,并求出$f(x_1)$;

步骤3 判断.

① 如果$|f(x_1)|<\varepsilon$,且$|x_1-x_0|<\varepsilon$或$|x_1-x_0|/|x_1|<\varepsilon$(其中$\varepsilon$是预先给定的精度),则迭代停止;否则,将$x_1$和$f(x_1)$代入迭代公式(2.26),重复步骤2和步骤3.

② 如果迭代的次数超过预先给定的最大迭代次数$gxmax$,则迭代停止,显示提示:'请注意:迭代次数超过给定的最大值 gxmax'.

已知方程 $f(x)=0$ 根的重数 m，求重根的修正牛顿切线法的 MATLAB 主程序

输入的量：初始值 x_0，方程 $f(x)=0$ 根 x^* 的重数 m，近似根 x_k 的误差限 tol，近似根 x_k 的函数值 $f(x_k)$ 误差限 $ftol$，迭代次数的最大值 $gxmax$、函数 $fnq(x)=f(x)$ 及其导数 $dfnq(x)=f'(x)$；

输出的量：迭代序列 $\{x_k\}$、迭代 k 次得到的迭代值 x_k（迭代次数达到最大值 $gxmax$ 时运行后输出信息'请注意：迭代次数超过给定的最大值 gxmax'）、相邻两次迭代的偏差 $piancha=|x_k-x_{k-1}|$ 和它的偏差的相对误差 $xdpiancha=|x_k-x_{k-1}|/|x_k|$ 的值.

根据式(2.26)和已知条件，编写一个名为 newtonxz.m 的 M 文件：

```
function [k,piancha,xdpiancha,xk,yk] =newtonxz(m,x0,tol,ftol,gxmax)
x(1) =x0;
for i =1: gxmax
x(i +1) =x(i) -m*fnq(x(i))/(dfnq(x(i)) +eps);
piancha =abs(x(i +1) -x(i));
xdpiancha =piancha/( abs(x(i +1)) +eps); i =i +1;
xk =x(i);yk =fnq(x(i));[(i -1) piancha xdpiancha xk yk]
  if ((piancha <tol) |(xdpiancha < tol))&(abs(yk) <ftol)
       k =i -1; xk =x(i);yk =fnq(x(i));[(i -1) piancha xdpiancha xk yk]
     return;
  end
end
if i >gxmax
    disp('请注意:迭代次数超过给定的最大值 gxmax.')
    k =i -1;xk =x(i);yk =fnq(x(i));[(i -1),piancha,xdpiancha,xk,yk];
    return;
end
```

例 2.6.7 判断 $x^*=0$ 是方程 $f(x)=2e^{3x}-9x^2-6x-2=0$ 的几重根？在区间 $[0,1]$ 上，分别用牛顿切线法和求重根的修正牛顿切线公式求此根的近似值 x_k，使其精确到 $\varepsilon=10^{-4}$.

解 (1) 先判断 $x^*=0$ 是已知方程的几重根.

$f(x)=2e^{3x}-9x^2-6x-2,\quad f(0)=0;$

$f'(x)=6e^{3x}-18x-6,\quad f'(0)=0;$

$f''(x)=18e^{3x}-18,\quad f''(0)=0;$

$f'''(x)=54e^{3x},\quad f'''(0)=54\neq 0.$

根据定理 2.6.1 知，$x^*=0$ 是方程 $f(x)=0$ 的三重根.

(2) 用牛顿切线法求近似根. 根据牛顿迭代公式

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}\quad (k=0,1,2,\cdots)$$

建立名为 fnq.m 和 dfnq.m 的 M 文件

```
function y = fnq(x)
e = exp(1); y = 2.*e.^(3.*x) - 9.*x.^2 - 6.*x - 2;
function y = dfnq(x)
e = exp(1); y = 6.*e.^(3.*x) - 18.*x - 6;
```

在 MATLAB 工作窗口输入程序

```
>> [k,xk,yk,piancha,xdpiancha] = newtonqx (0.5,1e-4, 1e-4,100)
```

运行后整理结果列入表 2-20.

表 2-20

k	*piancha*	*xdpiancha*	x_k	y_k
1	0.144 10	0.404 89	0.355 90	0.541 98
2	0.107 41	0.432 25	0.248 49	0.168 20
3	0.077 45	0.452 80	0.171 04	0.051 46
4	0.054 50	0.467 60	0.116 55	0.015 58
5	0.037 69	0.477 98	0.078 85	0.004 69
6	0.025 76	0.485 14	0.053 10	0.001 40
7	0.017 46	0.490 01	0.035 63	0.000 42
8	0.011 77	0.493 30	0.023 86	0.000 12
9	0.007 91	0.495 52	0.015 96	0.000 04
10	0.005 30	0.497 00	0.010 66	0.000 01
11	0.003 54	0.498 00	0.007 12	0.000 00
12	0.002 37	0.498 67	0.004 75	0.000 00
13	0.001 58	0.499 11	0.003 17	0.000 00
14	0.001 05	0.499 41	0.002 11	0.000 00
15	0.000 70	0.499 60	0.001 41	0.000 00
16	0.000 47	0.499 74	0.000 94	0.000 00
17	0.000 31	0.499 82	0.000 63	0.000 00
18	0.000 21	0.499 88	0.000 42	0.000 00
19	0.000 14	0.499 92	0.000 28	0.000 00
20	0.000 09	0.499 95	0.000 19	0.000 00

迭代次数 $k=20$,精确到 $\varepsilon=10^{-4}$的根的近似值是 $x_k=0.000\ 2$,其函数值是 $y_k=5.752\ 0e-011$,收敛速度较慢.

(3) 根据求重根的修正牛顿切线公式

$$x_{k+1}=x_k-m\frac{f(x_k)}{f'(x_k)},\quad (k=0,1,2,\cdots),$$

首先建立名为 fnq.m 和 dfnq.m 的 M 文件(见(2)),在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk,yk] = newtonxz(3,0.5,1e-4, 1e-4,100)
```

运行后整理结果得表 2-21.

表 2-21

k	*piancha*	*xdpiancha*	x_k	y_k
1	0.432 30	6.385 79	0.067 70	0.002 94
2	0.066 54	57.310 85	0.001 16	0.000 00
3	0.001 16	3443.447 27	0.000 00	0.000 00
4	0.000 43	1.000 78	-0.000 43	-0.000 00
5	0.000 43	9228.792 13	0.000 00	-0.000 00
6	0.011 15	1.000 00	0.011 15	0.000 01
7	0.011 12	356.869 17	0.000 03	0.000 00
8	0.000 03	1638.927 03	0.000 00	0.000 00

迭代次数 $k=8$,精确到 $\varepsilon=10^{-4}$的根的近似值是 $x_k=1.900\ 3e-008$,其函数值是 $y_k=4.440\ 9e-016$,*piancha* $=3.114\ 5e-005$,*xdpiancha* $=1638.927\ 0$,是二阶收敛(平方收敛). 可见,求重根的修正牛顿切线公式比牛顿切线法收敛速度快得多.

(二) 未知方程根的重数

方程 $f(x)=0$ 的根 x^* 的重数 m 未知时,求方程 $f(x)=0$ 的 m 重根的修正牛顿切线公式

$$x_{k+1}=x_k-\frac{u(x_k)}{u'(x_k)},\quad 其中\ u(x_k)=\frac{f(x_k)}{f'(x_k)}\ (k=0,1,2,\cdots) \tag{2.27}$$

产生的迭代序列 $\{x_k\}$ 是二阶收敛(平方收敛).

事实上,如果 x^* 是方程 $f(x)=0$ 的 m 重根,则 $f(x)=(x-x^*)^m g(x)$,且 $g(x^*)\neq 0$,则有

$$u(x)=\frac{f(x)}{f'(x)}=\frac{(x-x^*)^m g(x)}{m(x-x^*)^{m-1}g(x)+(x-x^*)^m g'(x)}$$

$$=\frac{(x-x^*)g(x)}{mg(x)+(x-x^*)g'(x)}$$

故 x^* 是方程 $u(x)$ 的单根.

因为修正牛顿切线迭代公式(2.27)的迭代函数为

$$\varphi(x)=x-\frac{u(x)}{u'(x)},\quad 其中\ u(x)=\frac{f(x)}{f'(x)}.$$

根据定理2.6可得,由迭代公式(2.27)产生的迭代序列 $\{x_k\}$ 是二阶收敛(平方收敛).

用迭代公式(2.27)求方程 $f(x)=0$ 的近似根 x_k 的一般步骤如下:

步骤1 选取初始值 x_0 和迭代的最大次数 $gxmax$ 以及计算精度 ε,计算 $f(x_0)$;

步骤2 用迭代公式(2.27)计算 x_1,并求出 $f(x_1)$;

步骤3 判断.

① 如果 $|f(x_1)|<\varepsilon$,且 $|x_1-x_0|<\varepsilon$ 或 $|x_1-x_0|/|x_1|<\varepsilon$(其中 ε 是预先给定的精度),则迭代停止;否则,将 x_1 和 $f(x_1)$ 代入迭代公式(2.27),重复步骤2和步骤3.

② 如果迭代的次数超过预先给定的最大迭代次数 $gxmax$,则迭代停止,显示提示:'请注意:迭代次数超过给定的最大值 gxmax'.

未知方程 $f(x)=0$ 根的重数为 m,求重根的修正牛顿切线法的MATLAB主程序

输入的量:初始值 x_0,近似根 x_k 的误差限 tol,近似根 x_k 的函数值 $f(x_k)$ 误差限 $ftol$,迭代次数的最大值 $gxmax$、函数 $fnq(x)=f(x)$ 及其一、二阶导数 $dfnq(x)=f'(x)$ 和 $ddfnq(x)=f''(x)$.

运行后输出的量:迭代序列 $\{x_k\}$、迭代 k 次得到的迭代值 x_k(迭代次数达到最大值 $gxmax$ 时,运行后输出信息'请注意:迭代次数超过给定的最大值 gxmax')、相邻两次迭代的偏差 $piancha=|x_k-x_{k-1}|$ 和它的偏差的相对误差 $xdpiancha=|x_k-x_{k-1}|/|x_k|$ 的值.

根据式(2.27)和已知条件,现编写一个名为 newtonxz1.m 的 M 文件

```
function [k,piancha,xdpiancha,xk,yk] = newtonxz1(x0,tol,ftol,
gxmax)
x(1) = x0;
for i = 1:gxmax
  u(i) = fnq(x(i))/dfnq(x(i));
  du(i) = 1 - fnq(x(i)) * ddfnq(x(i))/((dfnq(x(i)))^2 + eps);
```

```
    x(i+1)=x(i)-u(i)/du(i); piancha=abs(x(i+1)-x(i));
    xdpiancha=piancha/( abs(x(i+1))+eps);
    i=i+1; xk=x(i);yk=fnq(x(i));
    if ((piancha<tol)|(xdpiancha< tol))&(abs(yk)<ftol)
        k=i-1;xk=x(i);yk=fnq(x(i));[(i-1) piancha xdpiancha xk yk]
      return;
    end
  end
  if i>gxmax
        disp('请注意:迭代次数超过给定的最大值 gxmax.')
        k=i-1;xk=x(i);yk=fnq(x(i));[(i-1) piancha xdpiancha xk yk];
        return;
  end
```

例 2.6.8 用未知重数的求重根的修正牛顿切线法，求方程 $2e^{3x}-9x^2-6x-2=0$ 在区间[0,1]上的根，使精确到 $\varepsilon=10^{-4}$，并且与例 2.6.7 的结果比较.

解 (1) 建立并保存名为 fnq.m 和 dfnq.m (见例 2.6.7);

(2) 保存名为 newtonxz1.m 的 M 文件;

(3) 建立并保存名为 ddfnq.m

```
function y=ddfnq(x)
        e=exp(1); y=18*e^(3*x)-18;
```

(4) 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk,yk]=newtonxz1(0.5,1e-4, 1e-4,100)
```

运行后整理结果得表 2-22.

表 2-22

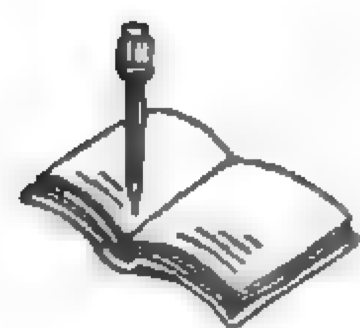
k	*piancha*	*xdpiancha*	x_k	y_k
1	0.599 23	6.038 57	-0.099 23	-0.008 18
2	0.096 98	43.054 85	-0.002 25	-0.000 00
3	0.002 25	1778.327 84	-0.000 00	0.000 00
4	0	0	-0.000 00	0

即迭代 $k=4$ 次，*piancha* =0，*xdpiancha* =0，$x_k=-1.265\,9e-006$，$y_k=0$. 这些结果与例 2.6.7 的结果比较见下表 2-23.

表 2-23

	牛顿切线法	已知重数的修正牛顿切线法	未知重数的修正牛顿切线法
迭代次数 k	20	7	4
根的近似值 x_k	1.855 7e-004	1.900 3e-008	-1.265 9e-006
偏差 *piancha*	9.277 5e-005	3.114 5e-005	0
相对误差 *xdpiancha*	4.999 5e-001	1638.927 0	0
x_k 的函数值 y_k	5.752 0e-011	4.440 9e-016	0
收敛速度	线性收敛	平方收敛	平方收敛

可见,未知重数的修正牛顿切线法的迭代次数和偏差最小;且根的精度最高;已知重数的修正牛顿切线法的迭代次数比牛顿切线法少 12 次,比未知重数的修正牛顿切线法多 4 次;三者中用牛顿切线法求得的近似根的误差最大.



习 题 2.6

1. 求方程 $e^x\sin x=2$ 的两个最小根,精确到 6 位有效数字. 如何确定初始值,初始值选取对收敛速度有何影响?

2. 求 $\sqrt{119}$ 的近似值,精确到 $\varepsilon=10^{-6}$.

3. 求下列方程在给定区间上的数值解,精确到 $\varepsilon=10^{-7}$.

(1) $x-\cos x=0$, $[0,1]$;　　(2) $x^3+2x^2+10x=20$, $[1,2]$.

4. 若 x^* 是 $f(x)=0$ 的单根,证明在牛顿切线法中

$$\lim_{k\to\infty}\frac{x_{k+1}-x^*}{(x_k-x^*)^2}=\frac{f''(x^*)}{2f'(x^*)}.$$

5. 判断 $x^*=0$ 是方程 $e^{2x}-1=2x(1+x)$ 的几重根? 在区间$[0,1]$上,分别用牛顿切线法和两种求重根的修正牛顿切线公式求此根的近似值 x_k,使精确到 $\varepsilon=10^{-3}$,并且进行比较.

6. 证明决定平方根$\sqrt{Q}$ $(Q\geqslant 0)$的公式

$$x_k=\frac{1}{2}\left(x_{k-1}+\frac{Q}{x_{k-1}}\right)\ (k=1,2,3,\cdots)$$

是牛顿切线迭代公式的一种特殊情况.

7. 求$\sqrt{3}$,精确到 $\varepsilon=10^{-9}$.

8. 导出找 Q 的 p 次根$\sqrt[p]{Q}$ $(Q\geqslant 0)$的迭代公式

$$x_k=x_{k-1}-\frac{x_{k-1}^p-Q}{px_{k-1}^{p-1}}\ (k=1,2,3,\cdots).$$

9. 应用第 8 题的迭代公式,寻找 2 的一个三次根,准确到 10^{-12}.

10. 用牛顿法求 $2\sin x = x^2$ 的两个最小实根,精确到 10^{-7},取不同的初始值计算,运行后输出根的近似值、近似值的函数值和迭代次数,分析两个根的收敛域;再用迭代法求解(可构造不同的迭代公式,如 $x=\sqrt{2\sin x}$ 等),进行比较.

11. 判断 $x^*=\sqrt{2}$是方程 $f(x)=x^4-4x^2+4=0$ 的几重根?在区间[1,2]上,分别用牛顿切线法和两种求重根的修正牛顿切线公式求此根的近似值 x_k,使精确到 $\varepsilon=10^{-14}$,并且进行比较.

12. 已知重数 m,证明:重根的修正牛顿切线公式

$$x_{k+1}=x_k-m\frac{f(x_k)}{f'(x_k)}\ (k=0,1,2,\cdots)$$

将产生二阶收敛(平方收敛)的迭代序列$\{x_k\}$.

2.7 割线法及其 MATLAB 程序

割线法又称**弦截法**、**弦位法**、**弦割法**和**弦法**.

牛顿切线法的突出优点是收敛速度快,但它也有明显的缺点:公式中含有导数,当$f(x)$较复杂时,使用不方便. 为了避免切线法计算导数的麻烦,我们现在设法利用一些函数值$f(x_k)$,$f(x_{k-1})$,…来回避导数值$f'(x_k)$的计算. 下面具体介绍割线法.

2.7.1 割线法

已知方程$f(x)=0$ 的两个近似根 x_k 和 x_{k-1},把牛顿迭代公式

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}\ (k=0,1,2,\cdots)$$

中的导数 $f'(x_k)$用差商 $\dfrac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$代替,整理后得

$$x_{k+1}=x_k-\frac{f(x_k)(x_k-x_{k-1})}{f(x_k)-f(x_{k-1})}\ (k=0,1,2,\cdots) \tag{2.28}$$

称**割线迭代公式**,用此公式求方程$f(x)=0$ 近似根的方法称**割线法**.

现在解释割线法的几何意义. 如图 2-24,曲线 $y=f(x)$上横坐标为 x_k 和 x_{k-1}的点分别记作 P_k 和 P_{k-1},则割线 P_kP_{k-1}的方程为

$$y-f(x_k)=\frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}(x-x_k)\ (k=0,1,2,\cdots).$$

令 $y=0$,即得到式(2.28),其中 x_{k+1}是割线 P_kP_{k-1}与 x 轴交点的横坐标,把 x_{k+1}作为根 x^* 的新的近似值. 过点 $P_{k+1}(x_{k+1},f(x_{k+1}))$和 P_k 作割线 P_kP_{k+1}交 x

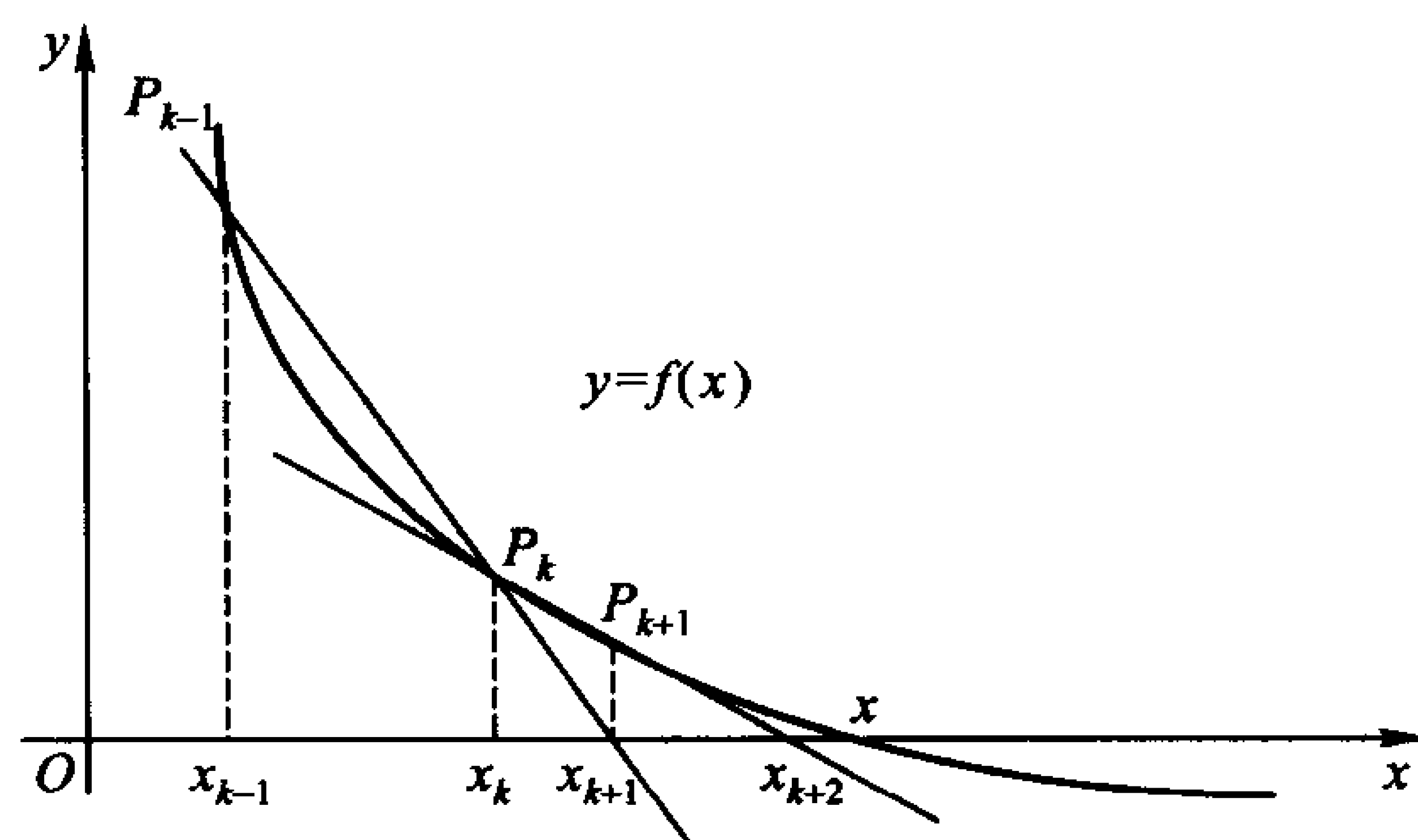


图 2-24 割线法的几何意义

轴于 x_{k+2},再把 x_{k+2} 作为根 x^* 的新的近似值. 如此继续下去,得到点列 $x_0,x_1,x_2,\cdots,x_k,\cdots$. 当此点列 $\{x_k\}$ 收敛到方程 $f(x)=0$ 的根 x^* 时,可求出满足精度要求的 x_k(见图 2-24). 这就是割线法名称的由来,可以看出它是**线性化与迭代法的结合**.

割线法的收敛速度比牛顿切线法稍慢(可以证明,割线法是局部收敛的,对于单根其收敛阶数是 1.618,),并且需要两个初始值 x_0,x_1 开始迭代.

2.7.2 割线法的 MATLAB 程序

(一) 用割线法求方程 $f(x)=0$ 的近似根 x_k 的一般步骤

步骤 1 选取初始值 x_0,x_1 和迭代的最大次数 $gxmax$ 以及计算精度 ε,计算 $f(x_0),f(x_1)$;

步骤 2 用割线迭代公式(2.28)计算 x_2,并求出 $f(x_2)$;

步骤 3 判断.

① 如果 $|f(x_2)|<\varepsilon$,且 $|x_2-x_1|<\varepsilon$ 或 $|x_2-x_1|/|x_2|<\varepsilon$ (其中 ε 是预先给定的精度),则迭代停止;否则,将 $x_2,x_1,f(x_2)$ 和 $f(x_1)$ 代入割线迭代公式(2.28),重复步骤 2 和步骤 3.

② 如果迭代的次数超过预先给定的最大次数 $gxmax$,则迭代停止,显示提示:'请注意:迭代次数超过给定的最大值 gxmax'.

(二) 割线法的 MATLAB 主程序

用割线法求方程 $f(x)=0$ 的近似根 x_k (精度为 tol)需要自行编制程序.

割线法的 MATLAB 主程序

输入的量：初始值 *x*01，*x*02，要求的近似根 x_k 的误差限 *tol*，x_k 的函数值 $f(x_k)$ 的误差限 *ftol*，迭代次数的最大值 *gxmax* 及其函数 *fnq*(x)；

输出的量：迭代序列 $\{x_k\}$、迭代 k 次得到的迭代值 x_k（迭代次数超过最大值 gxmax 时运行后输出信息'请注意：迭代次数超过给定的最大值 gxmax'）、相邻两次迭代的偏差 $piancha = |x_k - x_{k-1}|$ 和它的偏差的相对误差 $xdpiancha = |x_k - x_{k-1}| / |x_k|$ 的值.

使用初始值 x_0 和 x_1，根据式（2.28），现编写一个名为 gexian.m 的 M 文件：

```
function [k,piancha,xdpiancha,xk,yk] = gexian (x01,x02,tol,
ftol,gxmax)
  x(1) =x01;x(2) =x02;
  for i =2:gxmax
      u(i) = fnq(x(i)) * (x(i) -x(i -1)); v(i) = fnq(x(i)) - fnq
      (x(i -1));
      x(i +1) =x(i) - u(i)/( v(i)); piancha =abs(x(i +1) -x(i));
      xdpiancha = piancha/( abs(x(i +1)) +eps); i =i +1; xk = x(i);
        yk = fnq(x(i)); [(i -2) piancha xdpiancha xk yk]
    if (abs(yk) < ftol)&(( piancha <tol) |(xdpiancha < tol))
         k =i -2; xk =x(i);yk = fnq(x(i));[(i -2) piancha xdpi-
         ancha xk yk];
    return;
  end
  end
    if i >gxmax
  disp('请注意:迭代次数超过给定的最大值 gxmax.')
    k =i -z;xk =x(i);yk =fnq(x(i));[(i -2) piancha xdpiancha xk yk];
         return;
  end
```

（三）应用举例

例 2.7.1 用割线法求方程 $f(x) = e^{2x} - 3x^2 = 0$ 的一个实根的近似值 x_k，使精确到 $\varepsilon = 10^{-4}$.

解 （1）用作图法确定初始值 x_{01} 和 x_{02}.

在工作窗口输入程序

```
>> x = -2:0.0001:2; e =exp(1);
```

```
f=e.^(2.*x)-3.*x.^2;
plot(x,f),grid,
gtext('f=exp(2x)-3x^2')
```

运行后画出函数的图形(见图 2-25),可见在区间(-1,0)内有一个实根.取初始值 $x_{01}=-0.4$ 和 $x_{02}=-0.3$.

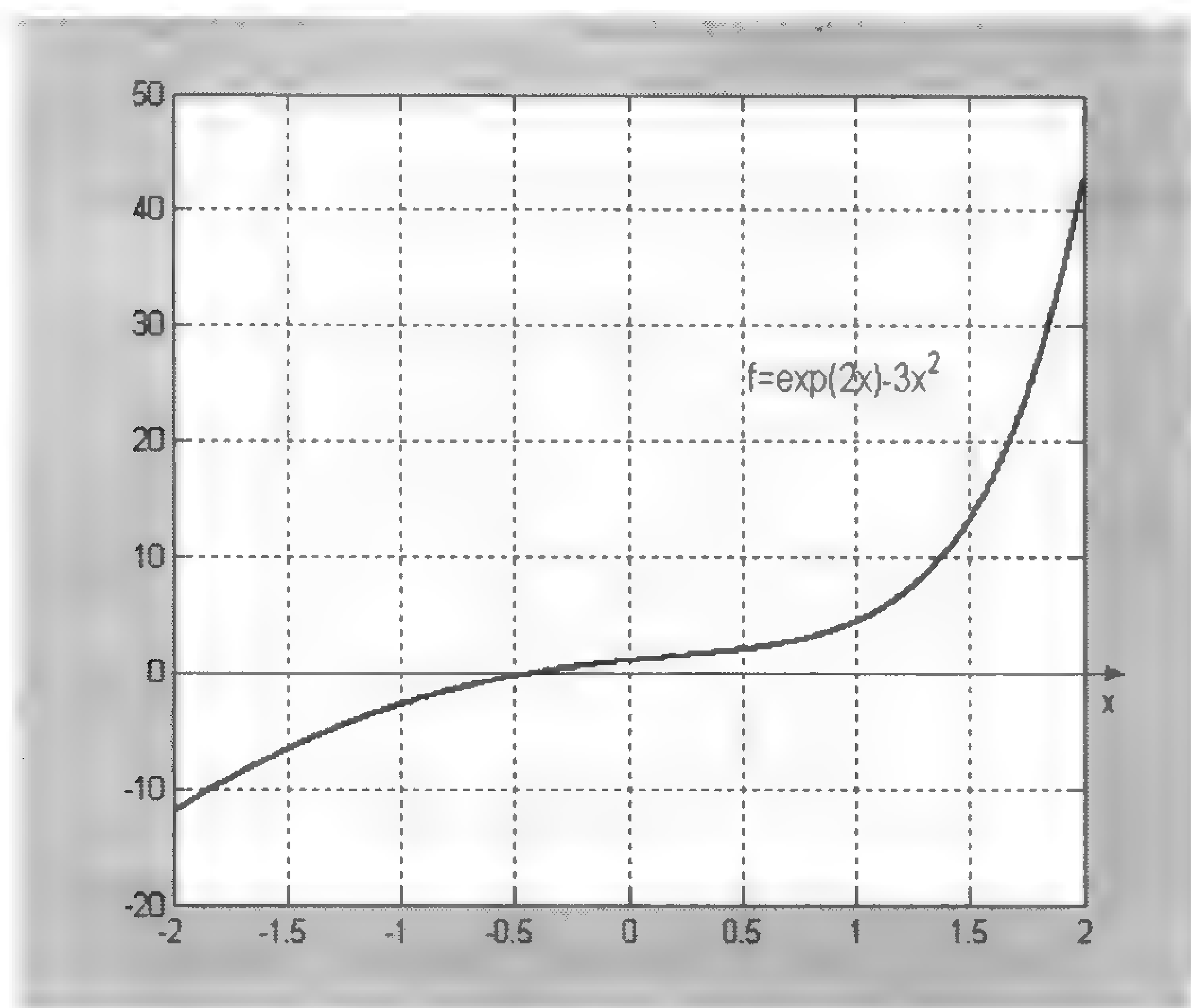


图 2-25 $f(x)=e^{2x}-3x^2$ 的图形

(2) 建立名为 fnq.m 的 M 文件

```
function f=fnq(x)
e=exp(1); f=e.^(2.*x)-3.*x.^2;
```

(3) 保存名为 gexian.m 的 M 文件;

(4) 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk,yk]=gexian(-0.4,-0.3,1e-4,1e-4,100)
```

(5) 运行后输出结果经整理得表 2-24.由此可见,迭代 $k=3$ 次,得到精度为 10^{-3} 的根的近似值 $x_k=-0.3906$,其函数值为 $y_k=3.8607e-008$,x_k 的相邻偏差为 $piancha=3.3271e-005$,其相对误差为 $xdpiancha=8.5168e-005$.

表 2－24

k	*piancha*	*xdpiancha*	x_k	y_k
1	0.090 09	0.230 95	−0.390 09	0.001 81
2	0.000 59	0.001 51	−0.390 68	−0.000 11
3	0.000 03	0.000 09	−0.390 65	0.000 00

例 2.7.2　门的气压控制

由汽缸控制关闭的门，关闭状态的示意图如图 2－26. 门宽 a，门枢在 H 处，与 H 相距为 b 处有一门销，通过活塞与圆柱形的汽缸相连，活塞半径 r，汽缸长 l_0，汽缸内气体的压强为 p_0. 当用力 F 推门，使门打开一个角度 α 时（示意图如图 2－27），活塞下降的距离为 c，门销与 H 的水平距离 b 保持不变，于是汽缸内的气体被压缩，对活塞的压强增加. 已知在绝热条件下，气体的压强 p 和体积 V 满足 $pV^t=c$，其中 t 是绝热系数，c 是常数. 试利用开门力矩和作用在活塞上的力矩相平衡的关系（对门枢而言），求在一定的力 F 作用下，门打开的角度 α. 设 $a=0.7\ \mathrm{m}, b=0.25\ \mathrm{m}, r=0.04\ \mathrm{m}, l_0=0.5\ \mathrm{m}, p_0=9999\ \mathrm{N/m^2}, t=1.4, F=25\ \mathrm{N}$.

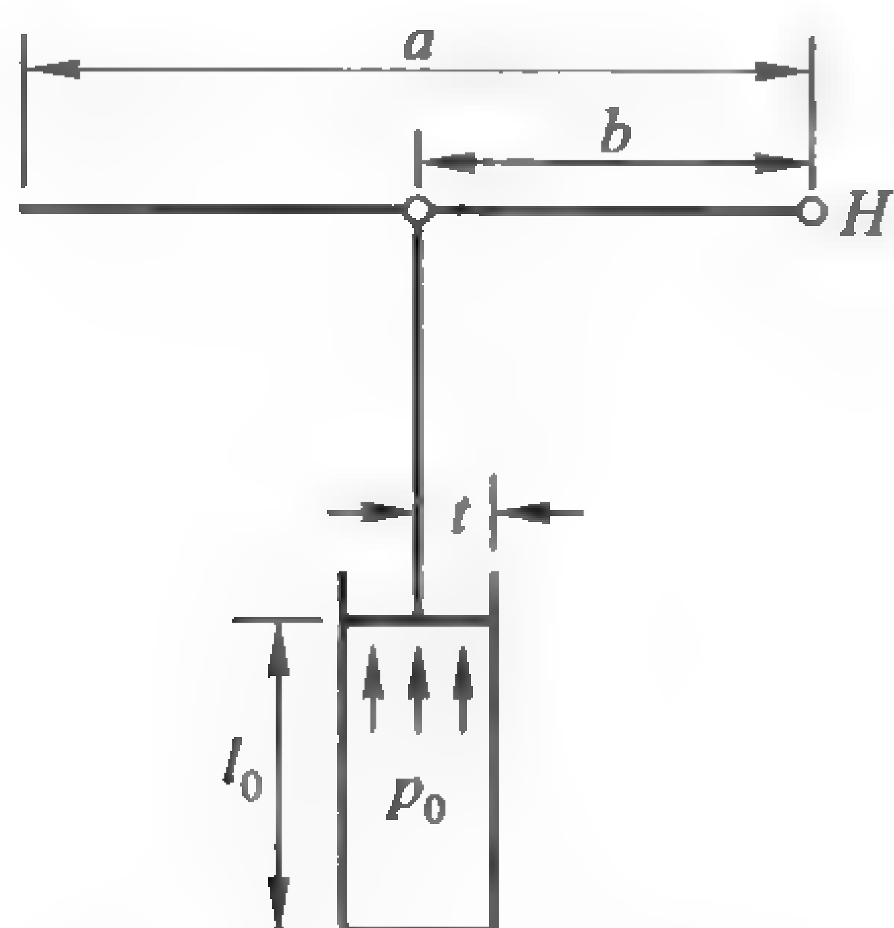


图 2－26　门的关闭状态

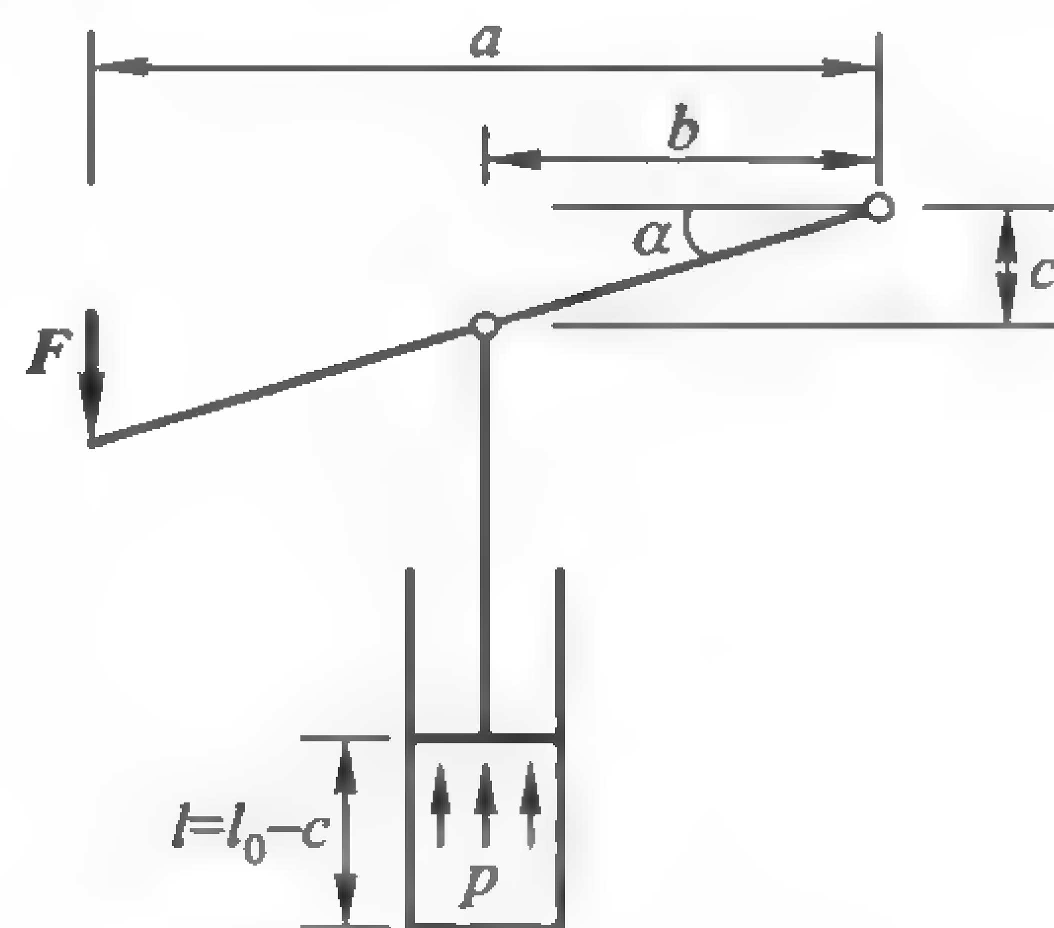


图 2－27　门的开启状态

解　首先利用力学和热学定律建立方程：

当在力 F 作用下，门打开的角度 α 时（如图 2－27），开门力矩（对门枢而言）为 $M_1=Fa\cos\alpha$.

当汽缸内的气体被压缩，气体的压强为 p，体积为 V，并记活塞的面积为 S，则气体作用在活塞上的力矩为 $M_2=pSb$. 力矩 M_1 与 M_2 相平衡，得

$$Fa\cos\alpha=pSb. \tag{2.29}$$

当门未开时（图 2－26）记汽缸内气体的压强为 p_0，体积为 V_0，根据绝热条件下，气体的压强和体积的关系，有 $pV^t=p_0V_0^t$，因为活塞的面积为不变，所以

$$p=p_0\left(\frac{V_0}{V}\right)^t=p_0\left(\frac{l_0}{l_0-c}\right)^t. \tag{2.30}$$

注意到 $c=b\cdot\tan\alpha, S=\pi r^2$，由(2.29)、(2.30)得

$$Fa\cos\alpha=\pi r^2 bp_0\left(\frac{l_0}{l_0-b\tan\alpha}\right)^t, \tag{2.31}$$

或表示为

$$f(\alpha)=Fa\cos\alpha(l_0-b\tan\alpha)^t-\pi r^2 bp_0 l_0^t=0, \tag{2.32}$$

这就是用于确定开门角度 α 的方程.

然后将已知量代入(2.32)式求非线性方程 $f(\alpha)=0$ 的近似解. 若用牛顿切线法需计算 $f(\alpha)$ 的导数，比较复杂，我们改按割线法(2.28)式的 $f(\alpha)$ 代入数据

$$a=0.7, b=0.25, r=0.04, l_0=0.5, p_0=9999, t=1.4, F=25,$$

编写 fnq.m 文件后保存

```
function f = fnq(x)
a =0.7; b =0.25; r =0.04; 10 =0.5; p0 =9999; t =1.4; F =25;
f =F * a * cos(x) * (10 -b * tan(x))^t  -pi * r^2 * b * p0 * 10^t;
```

任意给出 α 的两个初始值，输入

```
>> [k,piancha,xdpiancha,xk,yk] = gexian (0.2,0.4,1e -4,1e -4,100)
```

运行后得到结果

```
k =3,piancha =4.1014e -005,xdpiancha =1.2067e -004,
xk =0.3399,yk = -1.8581e -007
```

因此，$\alpha=0.339\,9$ 弧度 $=19.5°$.



习 题 2.7

1. 用割线法求方程 $2x=\cos x$ 在开区间 $(0,\pi/2)$ 内的实根 x^* 的近似值 x_k，使精确到 $\varepsilon=10^{-8}$.

2. 用割线法求方程 $f(\alpha)=Fa\cos\alpha(l_0-b\tan\alpha)^t-\pi r^2 bp_0 l_0^t=0$ 的实根 α 的近似值 x_k，使精确到 $\varepsilon=10^{-4}$，其中 $a=0.8, b=0.25, r=0.04, l_0=0.5, p_0=10\,000, t=1.4, F=25$.

3. 用割线法求方程 $xe^x-1=0$ 的实根 x 的近似值 x_k，使精确到 $\varepsilon=10^{-5}$.

4. 1 mol 理想气体的压强 p，体积 V，温度 T 满足关系 $pV=RT$，其中常数 $R=0.082\,05$ (atm/(K · mol))，而对于实际气体这个关系修正为 $\left(p+\frac{a}{V^c}\right)(V-b)=RT$，$a,b,c$ 为所给气体决定的常数. 现已知 $a=18.87, b=0.114\,2, c=2$，求气体在 $P=2$ atm，$T=315$ K 下的体积 V，使精确到 $\varepsilon=10^{-4}$.

2.8　抛物线法及其 MATLAB 程序

2.8.1　抛物线法

割线法是用过两点的一次多项式的零点近似替代方程 $f(x)=0$ 的根 x^*. 进一步可以用过三点的抛物线近似替代 $f(x)$，就是**抛物线法**，又称**米勒**(Müller)**法**. 由于抛物线法不需要计算导数 $f'(x_k)$ 的值，所以可以将它看成割线法的广义形式.

抛物线法是求方程 $f(x)=0$ 根 x^* 的近似值的一种迭代方法. 首先需要在根 x^* 的附近取三个初始值 x_0,x_1,x_2，设 x_2 是根 x^* 的最佳近似值. 利用它们对应的点 $(x_0,f(x_0))$，$(x_1,f(x_1))$，$(x_2,f(x_2))$ 构造一条抛物线

$$g(x)=a(x-x_2)^2+b(x-x_2)+c, \tag{2.33}$$

然后用二次函数 $g(x)$ 与 x 轴的交点之一作为根 x^* 的近似值 x_3，见图 2－28.

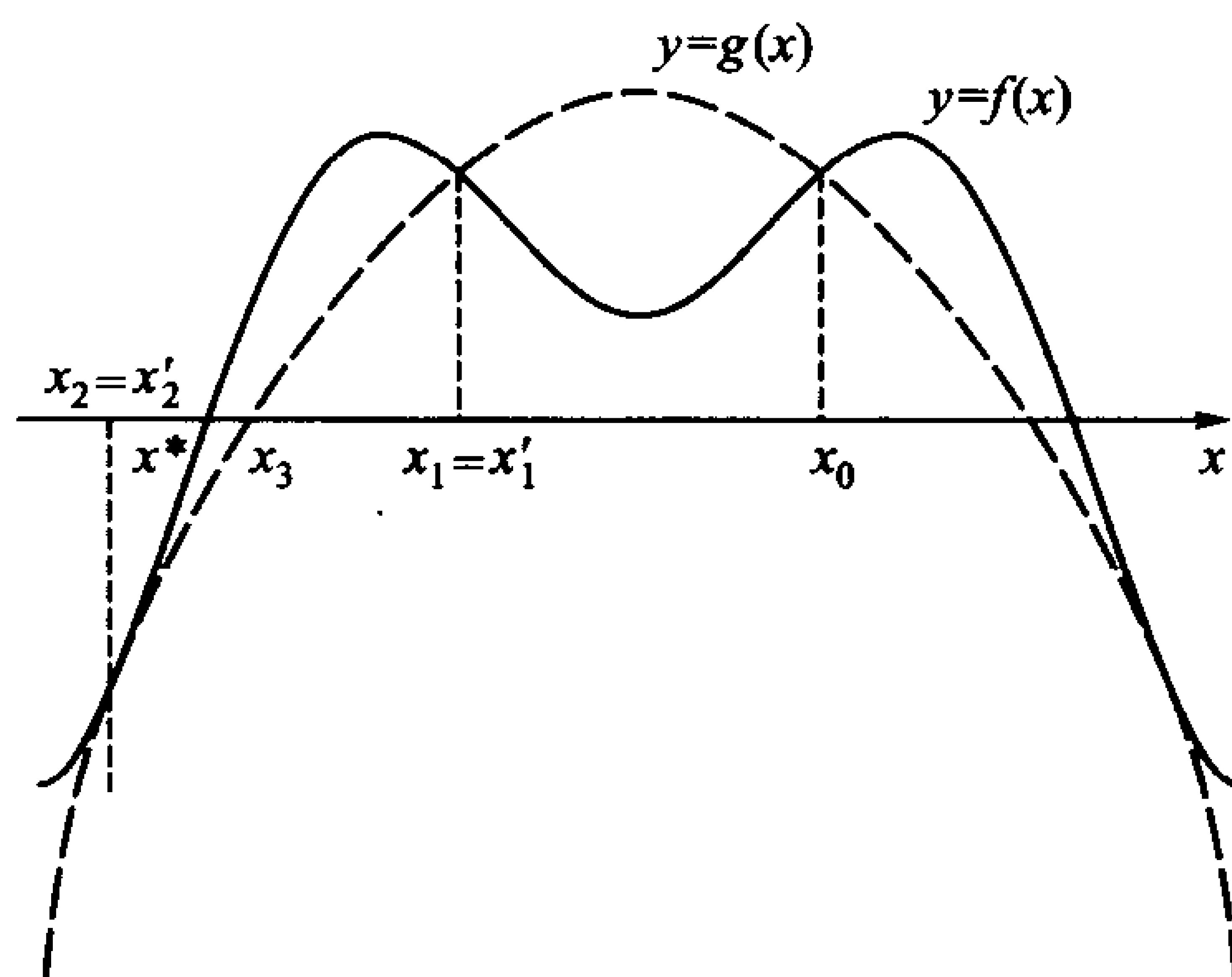


图 2－28

将 $(x_0,f(x_0))$，$(x_1,f(x_1))$，$(x_2,f(x_2))$ 分别代入(2.33)式，得到关于 a,b,c 的线性方程组

$$\begin{cases} a(x_0-x_2)^2+b(x_0-x_2)+c=f(x_0), \\ a(x_1-x_2)^2+b(x_1-x_2)+c=f(x_1), \\ a(x_2-x_2)^2+b(x_2-x_2)+c=f(x_2). \end{cases} \tag{2.34}$$

记 $f_0=f(x_0)$，$f_1=f(x_1)$，$f_2=f(x_2)$，

令

$$h_0=x_0-x_2,h_1=x_1-x_2,u_0=f_0-f_2,u_1=f_1-f_2, \tag{2.35}$$

则(2.34)式化为

$$\begin{cases} ah_0^2 + bh_0 + c = f_0, \\ ah_1^2 + bh_1 + c = f_1, \\ c = f_2. \end{cases}$$

解得

$$a = \frac{\begin{vmatrix} f_0 - c & h_0 \\ f_1 - c & h_1 \end{vmatrix}}{\begin{vmatrix} h_0^2 & h_0 \\ h_1^2 & h_1 \end{vmatrix}} = \frac{(f_0 - f_2)h_1 - (f_1 - f_2)h_0}{h_0^2 h_1 - h_1^2 h_0},$$

$$b = \frac{\begin{vmatrix} h_0^2 & f_0 - c \\ h_1^2 & f_1 - c \end{vmatrix}}{\begin{vmatrix} h_0^2 & h_0 \\ h_1^2 & h_1 \end{vmatrix}} = \frac{(f_1 - f_2)h_0^2 - (f_0 - f_2)h_1^2}{h_0^2 h_1 - h_1^2 h_0},$$

故

$$\begin{cases} a = \dfrac{u_0 h_1 - u_1 h_0}{h_0^2 h_1 - h_1^2 h_0}, \\ b = \dfrac{u_1 h_0^2 - u_0 h_1^2}{h_0^2 h_1 - h_1^2 h_0}, \\ c = f_2. \end{cases} \tag{2.36}$$

我们可以用求根公式 $x = x_2 + \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, (2.37)

即

$$x = x_2 + \frac{-2c}{b \pm \sqrt{b^2 - 4ac}} \tag{2.38}$$

求(2.33)的两个根 $x = x_1^*, x_2^*$. 因为 $c = f_2$,所以用(2.38)求根比用(2.37)求根简单.

为了确保方法的稳定性,需要选取(2.38)的右端第二项的模最小的根.即取

$$x_3 = \begin{cases} x_2 + \dfrac{-2c}{b + \sqrt{b^2 - 4ac}}, \text{当 } b > 0 \text{ 时}, \\ x_2 + \dfrac{-2c}{b - \sqrt{b^2 - 4ac}}, \text{当 } b < 0 \text{ 时}. \end{cases} \tag{2.39}$$

为了加速迭代,最好从 x_0, x_1, x_2 中取最靠近 x_3 的两个点作为新的 x_1', x_2',即放弃离 x_3 最远的点(见图 2-28),然后用 x_1', x_2', x_3 代替 x_0, x_1, x_2,重复以上过

程，直到误差满足要求为止.

可以证明当三个初始值在单根 x^* 附近时，抛物线法的收敛阶数是 1.84….抛物线法比割线法收敛速度快，它基本上与牛顿切线法一样快. 此方法可用来求方程的实数根和复数根. 我们还可以构造更高阶数的多项式方法，但是其实际意义不大.

用抛物线法求方程 $f(x)=0$ 近似根的步骤：

步骤 1 取三个初始值 x_0, x_1, x_2；

步骤 2 计算 $f(x_0), f(x_1), f(x_2)$，注意它们不全相等；

步骤 3 按公式(2.35)和(2.36)计算 $h_0, h_1, u_0, u_1, a, b, c$；

步骤 4 按公式(2.39)计算 x_3，并计算 $f(x_3)$；

步骤 5 判断.

如果 $|f(x_3)|<\varepsilon$，且 $|x_2-x_1|<\varepsilon$ 或 $|x_2-x_1|/|x_2|<\varepsilon$（其中 ε 是预先给定的精度），则停止计算，输出 $x^*\approx x_3$；否则，从 x_0, x_1, x_2 中取最靠近 x_3 的两个点作为新的 x_1', x_2'，然后用 x_1', x_2', x_3 代替 x_0, x_1, x_2，重复步骤 2 至步骤 5，直到误差满足要求为止.

2.8.2 抛物线法的 MATLAB 程序

用抛物线法求方程 $f(x)=0$ 的近似根 x_k（精度为 *tol*）需要自行编制程序.

抛物线法的 MATLAB 主程序

输入的量：初始值 x_1, x_2, x_3，要求的近似根 x_k 的误差限 *tol*，x_k 的函数值 $f(x_k)$ 的误差限 *ftol*，迭代次数的最大值 *gxmax* 及其函数 *fnq*(x)；

输出的量：迭代序列 $\{x_k\}$、迭代 k 次得到的迭代值 x_k、函数值 $f(x_k)$、相邻两次迭代的偏差 $piancha=\max\{|x_k-x_{k-1}|, |x_k-x_{k-2}|, |x_{k-1}-x_{k-2}|\}$ 和它的偏差的相对误差 $xdpiancha=piancha/|x_k|$ 的值（迭代次数超过最大值 *gxmax* 时运行后输出信息'请注意：迭代次数超过给定的最大值 gxmax，请重新输入初始值 x1,x2,x3 '）.

使用初始值 x_0, x_1 和 x_2，根据式按公式(2.35)，(2.36)和(2.39)，编写一个名为 paowu.m 的 M 文件

```
function [k,piancha,xdpiancha,xk,yk] = paowu(x1,x2, x3,tol,
ftol,gxmax)
  X = [x1, x2, x3];
  for i = 1:gxmax
      h0 = X(1) - X(3); h1 = X(2) - X(3);
      u0 = fnq(X(1)) - fnq(X(3));
```

```
    u1 = fnq (X (2)) - fnq (X (3));
    c = fnq (X (3));
    fenmu = h0^2 * h1 - h1^2 * h0; afenzi = u0 * h1 - u1 * h0;
    bfenzi = u1 * h0^2 - u0 * h1^2;
    a = afenzi/fenmu; b = bfenzi/fenmu;
    deta = b^2 - 4 * a * c;
    cha = [abs(X(3) - X(1)),abs(X(3) - X(2)),abs(X(2) - X(1))];
    piancha =min(cha); xdpiancha = piancha/( abs (X(3)) + eps);
  if deta < 0
     disp('提示:根的判别式值为负数.'),detakf = sqrt(deta);
  elseif deta < =0
     disp('提示:根的判别式值为零.'),detakf = 0;
  else
     disp('提示:根的判别式值为正数.'),detakf = sqrt(deta);
  end
  if b < 0
           detakf = - detakf;
  end
     z = -2 * c/(b + detakf);
     xk = X(3) + z;q1 = xk - X (1); q2 = xk - X (2);
     q3 = xk - X (3);
  if abs(q2) < abs(q1)
     X1 =[X(2), X(1), X(3)]; X = X1;Y = fnq(X);
  end
  if abs(q3) < abs(q1)
     X2 =[X(1), X(3), X(2)]; X = X2;Y = fnq(X);
  end
     X(3) = xk; Y(3) = fnq(X(3));
     yk = Y(3); [i piancha xdpiancha xk yk]
  if (abs(yk) < ftol)&(( piancha < tol) |(xdpiancha < tol))
          k = i; X(3) = xk; Y(3) = fnq(X(3));yk = Y(3);
       return;
    end
end
if i > gxmax
     disp('请注意:迭代次数超过给定的最大值 gxmax,请重新输入初始值
     x1,x2,x3 ')
  return;
```

```
end
  P=[i,piancha,xdpiancha,xk,yk]';
```

例 2.8.1 用抛物线法求方程 $f(x)=e^{2x}-3x^2=0$ 的一个实根的近似值 x_k，使精确到 $\varepsilon=10^{-4}$.

解 (1) 用作图法确定初始值 $x01$, $x02$ 和 $x03$

由例 2.7.1 的图 2-25，可见在区间 (-1,0) 内有一个实根，故取初始值 $x01=-0.4$, $x02=-0.3$ 和 $x03=-0.5$.

(2) 建立名为 fnq.m 的 M 文件(见例 2.7.1 的步骤(2))；

(3) 保存名为 gexian.m 的 M 文件；

(4) 在 MATLAB 工作窗口输入程序

```
>> [k,piancha,xdpiancha,xk,yk] = paowu ( -0.4, -0.3, -0.5,1e-
4, 1e-4,100)
```

(5) 运行后输出结果

```
提示:根的判别式值为正数.
ans =
  Columns 1 through 4
  1.00000000000000 0.10000000000000 0.20000000000000  -0.39066350562239
  Column 5
   -0.00005581900299
提示:根的判别式值为正数.
ans =
  Columns 1 through 4
  2.00000000000000 0.00933649437761 0.02389906977038  -0.39064638365394
  Column 5
   -0.00000000923532
提示:根的判别式值为正数.
ans =
  Columns 1 through 4
  3.00000000000000 0.00001712196845 0.00004382983990  -0.39064638082059
  Column 5
  0.00000000000000
k =
     3
piancha =
   1.712196845282676e-005
xdpiancha =
   4.382983989938760e-005
```

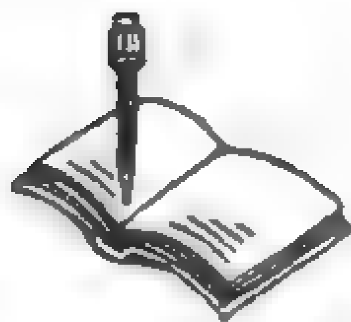
```
xk =
   -0.39064638082059
yk =
     2.220446049250313e-016
```

即,迭代 $k=3$ 次,得到精度为 10^{-4} 的根的近似值 $x_k=-0.3906$,其函数值为 $y_k=2.2204\mathrm{e}-016$,x_k 的相邻最小偏差为 *piancha* $=1.712\mathrm{e}-005$,其相对误差为 *xdpiancha* $=4.3830\mathrm{e}-005$.

将抛物线法与割线法的迭代结果进行比较,见表 2-25. 显然,抛物线法比割线法的迭代速度快.

表 2-25

迭代次数	抛物线法		割线法	
k	x_k	y_k	x_k	y_k
1.0000	-0.390 66	-0.000 06	-0.390 09	0.001 81
2.0000	-0.390 65	-0.000 00	0.001 51	-0.390 68
3.0000	-0.390 65	0.000 00	-0.390 65	0.000 00



习题 2.8

1. 用抛物线法求方程 $f(x)=2x^3-5x-1=0$ 的一个实根的近似值 x_k,使精确到 $\varepsilon=10^{-6}$.

2. 用抛物线法求方程 $f(x)=x^3-x+5=0$ 的全部实根,精确到 $\varepsilon=10^{-9}$.

3. 求曲线 $f(x)=x^2+2$ 与曲线 $g(x)=\dfrac{x}{5}-\sin x$ 之间的最小垂直距离处的 x 值,精确到小数点后 10 位.

2.9 求解非线性方程组的牛顿法及其 MATLAB 程序

求解非线性方程的牛顿法可以推广到二维非线性方程组的情形,也可以推广到更高维的情形.

2.9.1 求解二元非线性方程组的牛顿法及其 MATLAB 程序

(一) 求解二元非线性方程组的牛顿法

在实际问题中,经常遇到求解二元非线性方程组的问题. 例如,求圆 x^2+y^2

$=4$ 与双曲线 $x^2-y^2=1$ 的交点. 牛顿切线法求解非线性方程 $f(x)=0$ 的基本思想是将非线性方程作线性化后，通过建立迭代公式来求解. 这一方法也适用于解非线性方程组的情形.

一般的，设二元非线性方程组

$$\begin{cases} f_1(x,y)=0, \\ f_2(x,y)=0 \end{cases} \tag{2.40}$$

的解为 (x^*,y^*)，在 (x^*,y^*) 的某一邻域 $U(x^*,y^*)$ 内取一点 (x_k,y_k)，两函数 $f_1(x,y)$，$f_2(x,y)$ 在 $U(x^*,y^*)$ 内具有连续的各阶偏导数，则在邻域 $U(x^*,y^*)$ 内，$f_1(x,y)$，$f_2(x,y)$ 在点 (x_k,y_k) 的泰勒展开式为

$$\begin{aligned} f_i(x,y)=&f_i(x_k,y_k)+\left[(x-x_k)\frac{\partial}{\partial x}+(y-y_k)\frac{\partial}{\partial y}\right]f_i(x_k,y_k)+ \\ &\frac{1}{2!}\left[(x-x_k)\frac{\partial}{\partial x}+(y-y_k)\frac{\partial}{\partial y}\right]^2 f_i(x_k,y_k)+\cdots, \end{aligned} \tag{2.41}$$

其中 $i=1,2$. 去掉(2.41)式右端的 2 阶及 2 阶以上项(即线性化)后得

$$\begin{cases} f_1(x,y)\approx f_1(x_k,y_k)+(x-x_k)\dfrac{\partial f_1(x_k,y_k)}{\partial x}+(y-y_k)\dfrac{\partial f_1(x_k,y_k)}{\partial y}, \\ f_2(x,y)\approx f_2(x_k,y_k)+(x-x_k)\dfrac{\partial f_2(x_k,y_k)}{\partial x}+(y-y_k)\dfrac{\partial f_2(x_k,y_k)}{\partial y}. \end{cases} \tag{2.42}$$

将(2.40)的解 (x^*,y^*) 代入(2.42)式，可得

$$\begin{cases} (x^*-x_k)\dfrac{\partial f_1(x_k,y_k)}{\partial x}+(y^*-y_k)\dfrac{\partial f_1(x_k,y_k)}{\partial y}\approx -f_1(x_k,y_k), \\ (x^*-x_k)\dfrac{\partial f_2(x_k,y_k)}{\partial x}+(y^*-y_k)\dfrac{\partial f_2(x_k,y_k)}{\partial y}\approx -f_2(x_k,y_k). \end{cases} \tag{2.43}$$

将(2.43)式中的 x^*，y^* 分别用 x_{k+1}，y_{k+1} 代换后，两式的左、右端分别相等，即

$$\begin{cases} (x_{k+1}-x_k)\dfrac{\partial f_1(x_k,y_k)}{\partial x}+(y_{k+1}-y_k)\dfrac{\partial f_1(x_k,y_k)}{\partial y}=-f_1(x_k,y_k), \\ (x_{k+1}-x_k)\dfrac{\partial f_2(x_k,y_k)}{\partial x}+(y_{k+1}-y_k)\dfrac{\partial f_2(x_k,y_k)}{\partial y}=-f_2(x_k,y_k), \end{cases} \tag{2.44}$$

则(2.44)式是将非线性方程组(2.40)作线性化近似后得出的线性方程组. 当线性方程组(2.44)的系数行列式

$$D_k=\begin{vmatrix} \dfrac{\partial f_1(x_k,y_k)}{\partial x} & \dfrac{\partial f_1(x_k,y_k)}{\partial y} \\ \dfrac{\partial f_2(x_k,y_k)}{\partial x} & \dfrac{\partial f_2(x_k,y_k)}{\partial y} \end{vmatrix}\neq 0 \tag{2.45}$$

时，线性方程组(2.44)有唯一解：

$$\begin{cases} x_{k+1} = x_k - \dfrac{A(x_k,y_k)}{D_k}, \\ y_{k+1} = y_k - \dfrac{B(x_k,y_k)}{D_k} \end{cases} \quad (k=0,1,2,\cdots), \tag{2.46}$$

其中

$$\begin{cases} A(x_k,y_k) = \begin{vmatrix} f_1(x_k,y_k) & \dfrac{\partial f_1(x_k,y_k)}{\partial y} \\ f_2(x_k,y_k) & \dfrac{\partial f_2(x_k,y_k)}{\partial y} \end{vmatrix}, \\ B(x_k,y_k) = \begin{vmatrix} \dfrac{\partial f_1(x_k,y_k)}{\partial x} & f_1(x_k,y_k) \\ \dfrac{\partial f_2(x_k,y_k)}{\partial x} & f_2(x_k,y_k) \end{vmatrix}. \end{cases} \tag{2.47}$$

由(2.46)式计算出的 x_{k+1}, y_{k+1} 作为非线性方程组(2.40)的近似解,再代入(2.46)式中,依次进行解的迭代,直到相邻两次近似解满足条件:

$$\begin{cases} \max\left\{\dfrac{|x_{k+1}-x_k|}{|x_{k+1}|}, \dfrac{|y_{k+1}-y_k|}{|y_{k+1}|}\right\} < \varepsilon_1, \\ \text{或 } \max\{|x_{k+1}-x_k|, |y_{k+1}-y_k|\} < \varepsilon_1, \\ \text{且 } \max\{|f_1(x_{k+1},y_{k+1})|, |f_2(x_{k+1},y_{k+1})|\} < \varepsilon_2 \end{cases} \tag{2.48}$$

时为止. 用(2.46)式求解二元非线性方程组的方法称作**牛顿法**.

如果两个函数 $f_1(x,y)$,$f_2(x,y)$,及其二阶偏导数在解(x^*,y^*)的某一邻域 $U(x^*,y^*)$内连续且有界,$D \neq 0$,并且初始值选取得足够好,则由迭代公式(2.46)产生的迭代序列是二阶收敛的.

用牛顿法求解二元非线性方程组(2.40)近似根的步骤如下:

步骤 1 取两个初始值 x_0, y_0;

步骤 2 计算 $f_i(x_0,y_0)$, $\dfrac{\partial f_i(x_0,y_0)}{\partial x}$, $\dfrac{\partial f_i(x_0,y_0)}{\partial y}$,其中 $i=1,2$;

步骤 3 按公式(2.45)和(2.47)计算 $D_0, A(x_0,y_0), B(x_0,y_0)$;

步骤 4 当 $D_0 \neq 0$ 时,按公式(2.46)计算 x_1, y_1,并计算 $f_1(x_0,y_0)$, $f_2(x_0,y_0)$;

步骤 5 判断.

如果 $\max\{|f_1(x_1,y_1)|, |f_2(x_1,y_1)|\} < \varepsilon_2$,且

$$\max\{|x_1-x_0|, |y_1-y_0|\} < \varepsilon_1 \text{或} \max\left\{\dfrac{|x_1-x_0|}{|x_1|}, \dfrac{|y_1-y_0|}{|y_1|}\right\} < \varepsilon_1$$

(其中 $\varepsilon_1, \varepsilon_2$ 是预先给定的精度),则停止计算,输出 $x^* \approx x_1, y^* \approx y_1$;否则,计算

$\dfrac{\partial f_i(x_1,y_1)}{\partial x},\dfrac{\partial f_i(x_1,y_1)}{\partial y}$,其中 $i=1,2$. 用 $x_1,y_1,f_i(x_1,y_1),\dfrac{\partial f_i(x_1,y_1)}{\partial x},\dfrac{\partial f_i(x_1,y_1)}{\partial y}$代替 $x_0,y_0,f_i(x_0,y_0),\dfrac{\partial f_i(x_0,y_0)}{\partial x},\dfrac{\partial f_i(x_0,y_0)}{\partial y}$,其中 $i=1,2$,重复步骤 2 至步骤 5,直到误差满足要求为止.

说明:为了便于将二元非线性方程组推广到 n 元非线性方程组和编程方便,我们把迭代停止条件(2.48)改为

$$\begin{cases}\text{norm}(\boldsymbol{X}_{k+1}-\boldsymbol{X}_k)=\left[(x_{k+1}-x_k)^2+(y_{k+1}-y_k)^2\right]^{\frac{1}{2}}<\varepsilon_1,\\ \text{或}\dfrac{\text{norm}(\boldsymbol{X}_{k+1}-\boldsymbol{X}_k)}{\text{norm}(\boldsymbol{X}_{k+1})}=\dfrac{\left[(x_{k+1}-x_k)^2+(y_{k+1}-y_k)^2\right]^{\frac{1}{2}}}{\left[(x_{k+1})^2+(y_{k+1})^2\right]^{\frac{1}{2}}}<\varepsilon_1,\\ \text{且 norm}(\boldsymbol{F}_{k+1})=\left[(f_1(x_{k+1},y_{k+1}))^2+(f_2(x_{k+1},y_{k+1}))^2\right]^{\frac{1}{2}}<\varepsilon_2,\end{cases} \tag{2.49}$$

其中 $\text{norm}(\boldsymbol{X}_{k+1}-\boldsymbol{X}_k)=\left[(x_{k+1}-x_k)^2+(y_{k+1}-y_k)^2\right]^{\frac{1}{2}}$表示点 $A_{k+1}(x_{k+1},y_{k+1})$到点 $A_k(x_k,y_k)$的距离,也称作向量$(\boldsymbol{X}_{k+1}-\boldsymbol{X}_k)=\left\{x_{k+1}-x_k,y_{k+1}-y_k\right\}$的 2 范数,记作

$$\|\boldsymbol{X}_{k+1}-\boldsymbol{X}_k\|_2=\left[(x_{k+1}-x_k)^2+(y_{k+1}-y_k)^2\right]^{\frac{1}{2}}.$$

同理, $\|\boldsymbol{X}_{k+1}\|_2=\text{norm}(\boldsymbol{X}_{k+1})=\left[(x_{k+1})^2+(y_{k+1})^2\right]^{\frac{1}{2}}$ 即表示点 $A_{k+1}(x_{k+1},y_{k+1})$到原点 $O(0,0)$的距离,也表示向量 $\boldsymbol{X}_{k+1}=\left\{x_{k+1},y_{k+1}\right\}$的 2 范数.

$$\|\boldsymbol{F}_{k+1}\|_2=\text{norm}(\boldsymbol{F}_{k+1})=\left[(f_1(x_{k+1},y_{k+1}))^2+(f_2(x_{k+1},y_{k+1}))^2\right]^{\frac{1}{2}}$$

表示点 $B_{k+1}(x_{k+1},y_{k+1})$到原点 $O(0,0)$的距离,也称作向量 $\boldsymbol{F}_{k+1}=\left\{f_1(x_{k+1},y_{k+1}),f_2(x_{k+1},y_{k+1})\right\}$的 2 范数.

将向量的 2 范数的概念推广到 n 维向量 $\boldsymbol{\alpha}=\left\{x_1,x_2,\cdots,x_n\right\}$,即 n 维向量 $\boldsymbol{\alpha}$ 的 2 范数为

$$\|\boldsymbol{\alpha}\|_2=\text{norm}(\boldsymbol{\alpha})=(x_1^2+x_2^2+\cdots+x_n^2)^{\frac{1}{2}}.$$

(二)求解二元非线性方程组的牛顿法的 MATLAB 程序

1. MATLAB 主程序

用牛顿法求二元非线性方程组(2.40)的近似根 x_k,y_k(精度为 tol)需要自行编制程序.

解二元非线性方程组的牛顿法的 MATLAB 主程序

输入的量：初始值 $X=[x_0,y_0]$，要求的近似根 (x_k,y_k) 的误差限 *tol*，(x_k,y_k) 的函数值 $f_i(x_k,y_k)$，$i=1,2$ 的误差限 *ftol*，迭代次数的最大值 *ngmax* 和函数 $f_i(x,y)$，$i=1,2$ 及其一阶偏导数；

输出的量：迭代 $c_i=k$ 次得到的迭代向量值 $X=(x_k,y_k)$、向量的函数值 $Y=\{f_1(x_k,y_k),f_2(x_k,y_k)\}$、$Y$ 的 2 范数 *hanfan*、迭代方程组(2.44)的系数行列式 D 的值、相邻两次迭代向量的 2 范数 $danfan=\text{norm}(X_{k+1}-X_k)$ 和它的 2 范数的相对误差 $xddf=danfan/\text{norm}(X_{k+1})$ 的值.（当迭代次数超过最大值 *ngmax* 时运行后输出信息'请注意：迭代次数超过给定的最大值 ngmax，请重新输入初始值 x0，y0 '；当 $D=0$ 时，运行后输出信息'请注意！迭代方程组(2.44)的系数行列式的值等于零.'）

使用两个初始值 x_0,y_0，根据式按公式(2.45)，(2.46)，(2.47)和(2.49)，编写一个名为 newtonzu2.m 的 M 文件

```
function [ci,D,danfan,xddf,hanfan,Xk,Yk] = newtonzu2(X,tol,
ftol,ngmax)
  Y = Z(X);
for i = 1:ngmax
    dY = JZ(X);D = det(dY);
    A1 = [Y(1),dY(1,2); Y(2),dY(2,2)]; A = det(A1);
    B1 = [dY(1,1), Y(1); dY(2,1), Y(2)];B = det(B1);Xk = X - [A,B]/D;
    hanfan = norm(Y);danfan = norm(Xk - X);
    xddf = danfan/(norm(Xk) + eps);
    X = Xk;Y = Z(X);ci = i;
    if D ~ = 0
        ci = i; Xk = X - [A,B]/D;
        Yk = Y; [ci,D,danfan, xddf, hanfan , X, Y];
      else
      disp('请注意！迭代方程组(2.44)的系数行列式的值等于零.')
    end
    if (hanfan < ftol)&(( danfan < tol) |( xddf < tol))
      [ci,D,danfan, xddf, hanfan , X, Y];
      return;
    end
    if i > ngmax
      disp('请注意:迭代次数超过给定的最大值 ngmax,请重新输入初始值 x0,y0.')
      return;
```

```
    end
  end
```

2. 用解二元非线性方程组的牛顿法的 MATLAB 主程序求解二元非线性方程组(2.40)近似根的步骤

步骤 1 取两个初始值 x_0, y_0;

步骤 2 建立并保存名为 Z.m 的 M 文件;

```
function F = Z(X)
x = X(1);y = X(2); F = zeros(1,2);
F(1) = f1(x,y); F(2) = f2(x,y);
```

步骤 3 首先计算 $\frac{\partial f_1(x,y)}{\partial x} = df1x, \frac{\partial f_1(x,y)}{\partial y} = df1y, \frac{\partial f_2(x,y)}{\partial x} = df2x, \frac{\partial f_2(x,y)}{\partial y} = df2y$,然后建立并保存名为 JZ.m 的 M 文件

```
function dF = JZ(X)
x = X(1);y = X(2); dF = zeros(2,2); dF(1,1) = df1x;
dF(1,2) = df1y; dF(2,1) = df2x; dF(2,2) = df2y;
```

步骤 4 保存名为 newtonzu2.m 的 M 文件;

步骤 5 在 MATLAB 工作窗口输入程序

```
>> X = [ x0,y0];
[ci,D,danfan, xddf, hanfan , Xk, Yk] = newtonzu2 (X, tol,ftol,ng-
max)
```

步骤 6 运行后输出结果.

例 2.9.1 用迭代公式(2.46)求解方程组 $\begin{cases} x^2 + y^2 = 16, \\ x^2 - y^2 = 2. \end{cases}$

解 这里介绍两种求解已知方程组的方法.

方法 1 用解二元非线性方程组的牛顿法的 MATLAB 主程序求解.

(1) 设 $f_1(x) = x^2 + y^2 - 16$,

$f_2(x) = x^2 - y^2 - 2$,

则 $\frac{\partial f_1(x,y)}{\partial x} = 2x, \frac{\partial f_1(x,y)}{\partial y} = 2y, \frac{\partial f_2(x,y)}{\partial x} = 2x, \frac{\partial f_2(x,y)}{\partial y} = -2y.$

(2) 作函数 $f_1(x)$ 和 $f_2(x)$ 的图形(见图 2-29),取两个初始值 $x_0 = 2, y_0 = 2$.

(3) 建立并保存名为 Z.m 的 M 文件

```
function F = Z(X)
x = X(1);y = X(2); F = zeros(1,2);
F(1) = x^2 + y^2 - 16; F(2) - x^2 - y^2 - 2;
```

(4) 建立并保存名为 JZ.m 的 M 文件

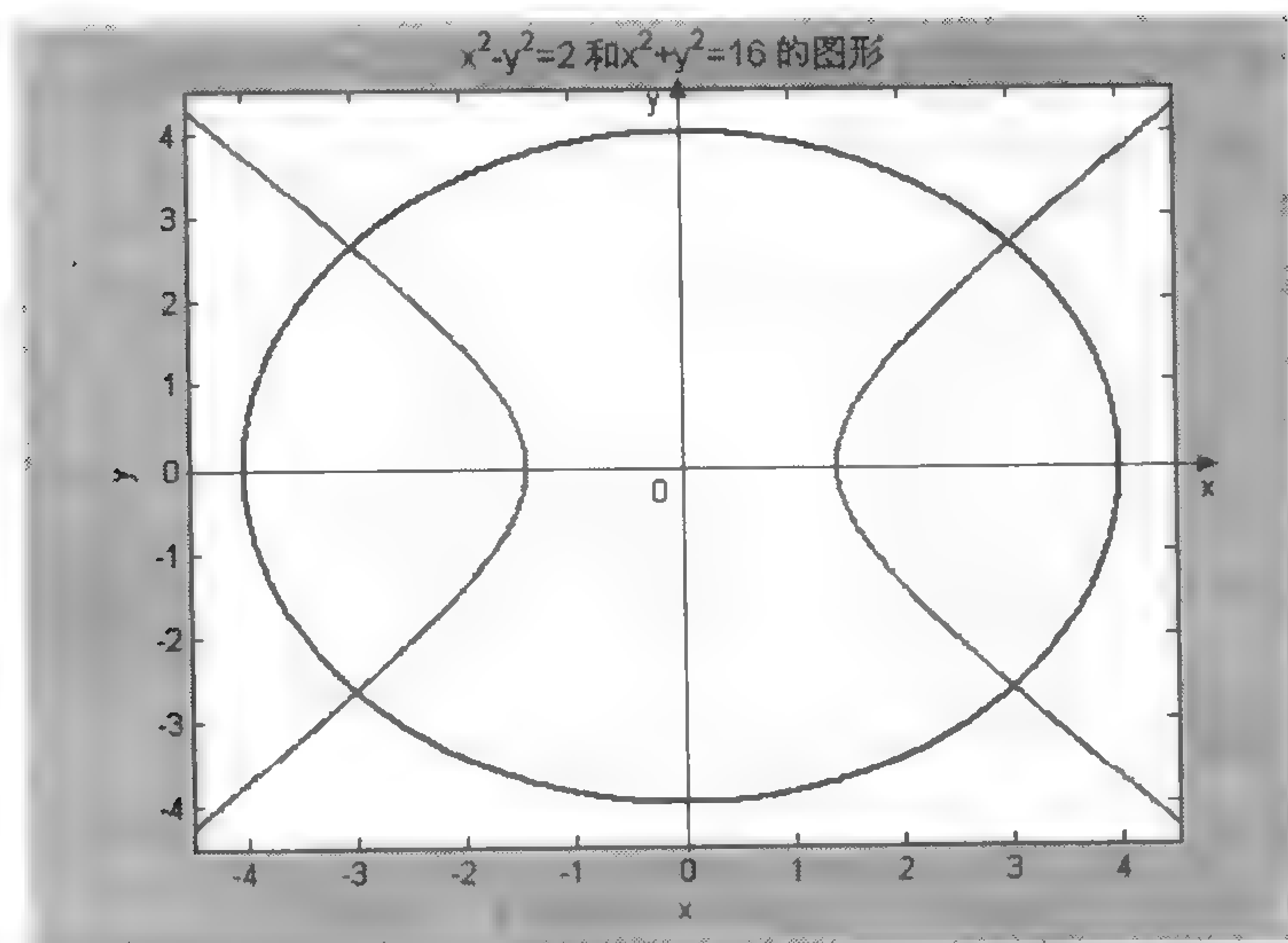


图 2-29

```
function dF = JZ(X)
        x = X(1);y = X(2); dF = zeros(2,2); dF(1,1) = 2 * x;
        dF(1,2) = 2 * y; dF(2,1) = 2 * x; dF(2,2) = -2 * y;
```

(5) 保存名为 newtonzu2.m 的 M 文件;

(6) 在 MATLAB 工作窗口输入程序

```
>> X = [2,2]; [k,D,danfan, xddf, hanfan , Xk, Yk] = newtonzu2 (X,
1e-4,1e-4,100)
```

(7) 运行后输出结果

```
k =
     5
D =
  -63.49803146638493
danfan =
     3.932201289951168e-011
xddf =
     9.830503224877920e-012
hanfan =
     3.336593498083849e-010
Xk =
```

```
    2.99999999996068   2.64575131106449
Yk =
  1.0e-015 *
     0  -0.88817841970013
```

方法 2 用解矩阵方程的方法求解.

编写如下程序

```
>> x0 =2;y0 =2; ngmax =10;tol =1e-6;x(1) =x0;y(1) =y0; i =1;u =
[1 1];k(1) =1;
while(norm(u) >tol*norm([x(i),y(i)]'))
  A =2*[x(i),y(i);x(i), -y(i)];
  b =[16 -x(i)^2 -y(i)^2,2 -x(i)^2 +y(i)^2]';
  u =A\b;x(i +1) =x(i) +u(1); y(i +1) =y(i) +u(2); i =i +1;k(i) =i;
  if(i > ngmax)
     error('请注意:迭代次数超过给定的最大值 ngmax,请重新输入初始值');
  end
end
 [k',x',y']
```

运行后输出结果,取初始值 $\boldsymbol{x}=(2,2)^{\mathrm{T}}$,迭代次数 $n=6$,精度 $\varepsilon=10^{-6}$,$x_6=3.000\,000$,$y_6=2.645\,751$ 与精确解的误差已不超过 ε.

2.9.2 求解 n 元非线性方程组的牛顿法及其 MATLAB 程序

(一) 求解 n 元非线性方程组的牛顿法

一般的,方程组记作

$$f(\boldsymbol{x})=0, \tag{2.50}$$

其中 $\boldsymbol{x}=(x_1,\cdots,x_n)^{\mathrm{T}}$, $\boldsymbol{f}(\boldsymbol{x})=(f_1(\boldsymbol{x}),\cdots,f_n(\boldsymbol{x}))^{\mathrm{T}}$. 设 $\boldsymbol{x}^{(k)}=(x_1^{(k)},\cdots,x_n^{(k)})^{\mathrm{T}}$ 是方程组(2.50)的第 k 步近似解,与 2.9.1 中的牛顿法类似,在 $\boldsymbol{x}^{(k)}$ 作泰勒展开,线性化后用 $\boldsymbol{x}^{(k+1)}$ 代替 $\boldsymbol{x}$ 可得

$$f_i(\boldsymbol{x}^{(k+1)})=f_i(\boldsymbol{x}^{(k)})+\frac{\partial f_i(\boldsymbol{x}^{(k)})}{\partial x_1}(x_1^{(k+1)}-x_1^{(k)})+\cdots+\frac{\partial f_i(\boldsymbol{x}^{(k)})}{\partial x_n}(x_n^{(k+1)}-x_n^{(k)})\quad(i=1,2,\cdots,n), \tag{2.51}$$

记

$$f'(x)=\begin{pmatrix}\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n}\\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n}\\ \vdots & \vdots & & \vdots\\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n}\end{pmatrix} \tag{2.52}$$

称(2.52)为f的**雅可比矩阵**.(2.51)式可写作

$$f(x^{(k+1)})=f(x^{(k)})+f'(x^{(k)})(x^{(k+1)}-x^{(k)}). \tag{2.53}$$

若$f'(x^{(k)})$可逆,则由(2.53)可得求解方程组(2.50)的牛顿迭代公式

$$x^{(k+1)}=x^{(k)}-\left[f'(x^{(k)})\right]^{-1}f(x^{(k)}). \tag{2.54}$$

实际上在计算过程的第k步,先计算$f(x^{(k)})$和$f'(x^{(k)})$,再解方程组

$$f'(x^{(k)})\Delta x^{(k)}=-f(x^{(k)}), \tag{2.55}$$

得到$\Delta x^{(k)}$后,令

$$x^{(k+1)}=x^{(k)}+\Delta x^{(k)} \tag{2.56}$$

即可.

有定理表明,牛顿迭代公式(2.54)是超线性收敛的,稍加条件就至少是平方收敛的.

(二) 求解 n 元非线性方程组的牛顿法的 MATLAB 程序

用牛顿法求n元非线性方程组(2.50)的近似根$x^{(k)}$(精度为tol)需要自行编制如下两个调用函数程序和主程序.

1. 建立调用函数程序 1

根据非线性方程组(2.50)编写调用函数程序的 M 文件,取名为 Z.m:

```
function F = Z(X)
x1 = X(1);x2 = X(2); …;xn = X(n); F = zeros(1,n);
F(1) = f1(X); F(2) = f2(X); …;F(n) = fn(X);
```

2. 建立调用函数程序 2

首先计算$\frac{\partial f_1(X)}{\partial x_1}=df1x1,\cdots,\frac{\partial f_1(X)}{\partial x_n}=df1xn,\frac{\partial f_2(X)}{\partial x_1}=df2x1,\cdots,\frac{\partial f_2(X)}{\partial x_n}=df2xn,\cdots,\frac{\partial f_n(X)}{\partial x_1}=dfnx1,\cdots,\frac{\partial f_n(X)}{\partial x_n}=dfnxn$,然后根据$f$的雅可比矩阵(2.52)编写调用函数程序的 M 文件,取名为 Z.m :

```
function dF = JZ(X)
    x1 = X(1);x2 = X(2); …; xn = X(n); dF = zeros(n,n);
    dF(1,1) = df1x1; …; dF(1,n) = df1xn;
```

```
dF(2,1) = df2x1; …; dF(2,n) = df2 xn;
…………
dF(n,1) = dfnx1; …; dF(n,n) = dfnxn;
```

3. 建立 MATLAB 主程序

解 n 元非线性方程组的牛顿法的 MATLAB 主程序

输入的量：初始值 X_0，要求的近似根 $x^{(k)}$ 的误差限 tol，$x^{(k)}$ 的函数值 $f_i(x^{(k)})$，$i=1,2,\cdots,n$ 的误差限 $ftol$，迭代次数的最大值 $gxmax$ 和函数 $f_i(x^{(k)})$，$i=1,2,\cdots,n$ 及其一阶偏导数；

输出的量：迭代 k 次数得到的迭代向量值 $x^{(k)}$、向量的函数值 $f_i(x^{(k)})$ $i=1,2,\cdots,n$ 及其 2 范数 $hanfan$、迭代方程组(2.51)的雅可比矩阵(2.52)的行列式 D 的值、相邻两次迭代向量的 2 范数 $danfan=norm(x^{(k+1)}-x^{(k)})$ 和它的 2 范数的相对误差 $xddf=danfan/norm(x^{(k+1)})$ 的值.（当迭代次数超过最大值 $gxmax$ 时运行后输出信息'请注意：迭代次数超过给定的最大值 gxmax，请重新输入初始值'；当 $D=0$ 时，运行后输出信息'请注意！ 迭代方程组(2.51)的系数行列式的值等于零.'）

使用两个初始值 x_0，y_0，根据式按公式(2.45)，(2.46)，(2.47)和(2.49)，编写一个名为 newtonzu2.m 的 M 文件

```
function [ci,D,danfan, xddf, hanfan , Xk, Yk] = newtonzun (X, tol, ftol,gxmax)
    Y = Z(X);
    for i = 1:gxmax
      dY = JZ(X);D = det(dY);Xk = X - (dY \ Y')';
      hanfan = norm(Y);danfan = norm(Xk - X);
      xddf = danfan/(norm(Xk) + eps);X = Xk;Y = Z(X);ci = i;
      if D ~ =0
        ci = i; Xk = X - (dY \Y')';Yk = Y;[ci,D,danfan,xddf,hanfan,X,Y];
      else
        disp('请注意！ 迭代方程组(2.51)的系数行列式的值等于零.')
      end
      if (hanfan < ftol)&(( danfan < tol) |( xddf < tol))
        [ci,D,danfan, xddf, hanfan , X, Y];
        return;
      end
    end
    if i > gxmax
      disp('请注意：迭代次数超过给定的最大值 gxmax,请重新输入初始值.')
```

```
  return;
end
```

4. 用解 n 元非线性方程组的牛顿法的 MATLAB 主程序求解 n 元非线性方程组(2.50)近似根的步骤与二元类似.

例 2.9.2 用迭代公式(2.54)求解方程组 $\begin{cases} x^2+y^2=16, \\ x^2-y^2=2, \end{cases}$ 要求精度 $\varepsilon=10^{-6}$.

解 这里介绍两种求解已知方程组的方法.

方法 1 用解 n 元非线性方程组的牛顿法的 MATLAB 主程序求解

(1)至(4)步骤与例 2.9.1 完全相同;

(5) 保存名为 newtonzun.m 的 M 文件;

(6) 在 MATLAB 工作窗口输入程序

```
>> X=[2,2];
 [k,D,danfan, xddf, hanfan , Xk, Yk] = newtonzun (X,1e-6,1e-6,
100)
```

(7) 运行后输出结果

```
k=5,D= -63.49803146638493,danfan=3.932201289951168e-011
xddf=9.830503224877920e-012,hanfan=3.336593498083849e-010
Xk = 3.00000000000000 2.64575131106459
Yk - 1.0e-015 *
  0 -0.88817841970013
```

方法 2 用解矩阵方程的方法求解

(1) 在 MATLAB 工作窗口输入程序

```
>> x0=2;y0=2; gxmax =10;tol=1e-6;x(1)=x0;y(1)=y0;i=1;u=
[1 1];k(1)=1;
while(norm(u)>tol*norm([x(i),y(i)]'))
  A=2*[x(i),y(i);x(i),-y(i)];
  b=[16-x(i)^2-y(i)^2,2-x(i)^2+y(i)^2]';
  u=A\b; x(i+1)=x(i)+u(1); y(i+1)=y(i)+u(2); i=i+1;
  k(i)=i;
  if(i > gxmax)
  error('请注意:迭代次数超过给定的最大值 gxmax,请重新输入初始值');
  end
end
[k',x',y']
```

(2) 运行后输出结果

```
ans =
   1.00000000000000   2.00000000000000   2.00000000000000
```

```
2.00000000000000  3.25000000000000  2.75000000000000
3.00000000000000  3.00961538461538  2.64772727272727
4.00000000000000  3.00001536003932  2.64575204838080
5.00000000000000  3.00000000003932  2.64575131106469
6.00000000000000  3.00000000000000  2.64575131106459
```

即,取初始值 $x=(2,2)^{\mathrm{T}}$,迭代次数 $n=6$,精度 $\varepsilon=10^{-6}$,$x_6=3.000\ 000$,$y_6=2.645\ 751$与精确解的误差已不超过 ε.

2.9.3 求解 n 元非线性方程组的拟牛顿法及其 MATLAB 程序

(一)求解 n 元非线性方程组的拟牛顿法

用公式(2.54)求解方程组(2.50)时,每一步都要计算雅可比矩阵 $\boldsymbol{f}'(\boldsymbol{x}^{(k)})$,当函数 f 比较复杂时显然是不方便的.希望能用较简单的矩阵 $\boldsymbol{A}^{(k)}$代替$\boldsymbol{f}'(\boldsymbol{x}^{(k)})$,将(2.54)式变为

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-(\boldsymbol{A}^{(k)})^{-1}\boldsymbol{f}(\boldsymbol{x}^{(k)}). \tag{2.57}$$

仿照2.7节解一元非线性方程的割线法中用插商$\dfrac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$代替 $f'(x_k)$的作法,使 $\boldsymbol{A}^{(k)}$满足

$$\boldsymbol{A}^{(k)}=\frac{\boldsymbol{f}(\boldsymbol{x}^{(k)})-\boldsymbol{f}(\boldsymbol{x}^{(k-1)})}{\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k-1)}}. \tag{2.58}$$

但是(2.58)式不足以确定 $\boldsymbol{A}^{(k)}$,通常再用迭代

$$\boldsymbol{A}^{(k+1)}=\boldsymbol{A}^{(k)}+\Delta\boldsymbol{A}^{(k)} \tag{2.59}$$

计算之.(2.57)—(2.59)所采用的方法称为**拟牛顿法**.至于(2.59)式中的 $\Delta\boldsymbol{A}^{(k)}$又有不同的构造方法,有兴趣的读者可参看其他数值分析书.

(二)拟牛顿法的 MATLAB 程序

用拟牛顿法求 n 二元非线性方程组(2.50)的近似根 $\boldsymbol{x}^{(k)}$(精度为 *tol*),需要根据(2.50)式和(2.58)式分别自行编制名为 NN.m 和 NF.m 的 M 文件作为调用函数程序,然后根据(2.57)式,编写名为 ninewton.m 的主程序

```
function [k,hanfan,danfan,xddf,Xk,Yk] = ninewton(X01,X02,tol,
ftol,gxmax)
   X = [X01;X02];n = length(X);
   for j =1:gxmax
     Y = NN(X); A = NF(X);D = det(A); Xk = X - ((A + ones(n) * eps) \ Y')';
     hanfan = norm(Y(2));danfan = norm(Xk(2) - X(2));
     xddf = danfan/(norm(Xk(2)) + eps);j = j +1;
     X = Xk;Y = Z(X);Yk = Y;k = j;warning off MATLAB:singularMatrix
     if D ~ =0
```

```
                Xk = X - (A \Y ')';Yk = Y;j = j + 1; [j - 1,D,danfan, xd-
                df, hanfan ,Xk, Yk]'
        else
          disp('请注意！迭代方程组(2.59)中的矩阵(2.58)行列式的值等于零.')
        end
        if (hanfan < ftol) |(( danfan < tol) |( xddf < tol))
            [j -1,D,danfan, xddf, hanfan ,Xk, Yk]';
            warning off MATLAB:singularMatrix
          return;
        end
      end
      if(j > gxmax)
        error('请注意:迭代次数超过给定的最大值 gxmax.');
      end
```

其中输入的量:初始值 X_{01}, X_{02},要求的近似根 $x^{(k)}$ 的误差限 tol, $x^{(k)}$ 的函数值 $f_i(x^{(k)})$, $i=1,2,\cdots,n$ 的误差限 $ftol$,迭代次数的最大值 $gxmax$ 和函数 $f_i(x^{(k)})$, $i=1,2,\cdots,n$ 及其一阶偏导数.

运行后输出的量:迭代 k 次得到的迭代向量值 $x^{(k)}$、向量的函数值 $f_i(x^{(k)})$, $i=1,2,\cdots,n$ 及其 2 范数 $hanfan$、迭代方程组(2.51)的雅可比矩阵(2.52)的行列式 D 的值、相邻两次迭代向量的 2 范数 $danfan=\text{norm}(x^{(k+1)}-x^{(k)})$ 和它的 2—范数的相对误差 $xddf=danfan/\text{norm}(x^{(k+1)})$ 的值.(当矩阵(2.58)行列式的值等于零时,输出信息'请注意！迭代方程组(2.59)中的矩阵(2.58)行列式的值等于零.';当迭代次数超过最大值 $gxmax$ 时运行后输出信息 '请注意:迭代次数超过给定的最大值 gxmax,请重新输入初始值').



习 题 2.9

1. 考虑如何用牛顿法和拟牛顿法解非线性方程组 $x^2+y^2=4$, $x^2-y^2=1$, 精确到 $\varepsilon=10^{-6}$.

2. 用牛顿法解非线性方程组 $x^2+4y^2=4$, $2x^2-4x-2y=-1$, 取初始值 $(x_0,y_0)=(2,0.25)$, 精确到 $\varepsilon=10^{-6}$.

3. 证明方程 $x+\sin x=1$ 在 $(0,1)$ 内有一个实根,用二分法求误差不大于 0.5×10^{-4} 的根,需要迭代多少次?

4. 用两种方法求方程 $x^{11}-12x^8+x^5-3x^2-4=0$ 的精确解.

5. 利用作图法判断方程 $e^x=2-10x$ 是否有正根,如果有,请确定正根所在的区间,并且用二分法、逐步搜索法和迭代法求之,精确到 $\varepsilon=10^{-3}$.

6. 用两种方法判断下列方程是否有实根，如果有，请确定其隔根区间，并且用五种方法求之，精确到 $\varepsilon = 10^{-4}$.

(1) $e^{-x} = 2 - x$; (2) $x^3 = 5x^2 + 3$.

7. 用割线法、牛顿切线法和抛物线法

(1) 求方程 $f(x) = x^3 - 3x^2 - x + 9 = 0$ 的全部实根的近似值 x_k，使精确到 $\varepsilon = 10^{-6}$;

(2) 求方程 $f(x) = 2\cos x - 2\sin x - 1 = 0$ 的最小正根的近似值 x_k，精确到 $\varepsilon = 10^{-9}$;

(3) 求方程 $f(x) = 3\cos x - 10x + 10 = 0$ 的全部实根的近似值 x_k，使精确到 $\varepsilon = 10^{-4}$.

8. 用迭代公式 $x_{k+1} = g(x_k)$ 计算序列 $\{x_k\}$，分析其收敛性，其中 $g(x)$ 为(任选其一)：

(1) $g(x) = rx(1-x)$，r 分别取 1.7, 2.8, 3.3, 3.5, 3.6，初始值 $0 < x_0 < 1$;

(2) $g(x) = ax\,e^{-bx}$，a 分别取 5, 11, 15，$b\ (>0)$ 任意，初始值 $x_0 = 1$.

9. 就下列函数讨论牛顿切线法的收敛性和收敛速度：

(1) $f(x) = \begin{cases} \sqrt{x}, & x \geqslant 0, \\ -\sqrt{-x}, & x < 0; \end{cases}$ (2) $f(x) = \begin{cases} \sqrt[3]{x^2}, & x \geqslant 0, \\ -\sqrt[3]{x^2}, & x < 0. \end{cases}$

10. 水槽由半圆柱体水平放置而成，如图 2-30. 圆柱体长 L，半径 r，当给定水槽内盛水的体积 V 后，要求计算从水槽边沿到水面的距离 x. 今已知 $L = 25.4$ m，$r = 2$ m，求 V 分别为 10 m^3, 50 m^3, 100 m^3 的 x.

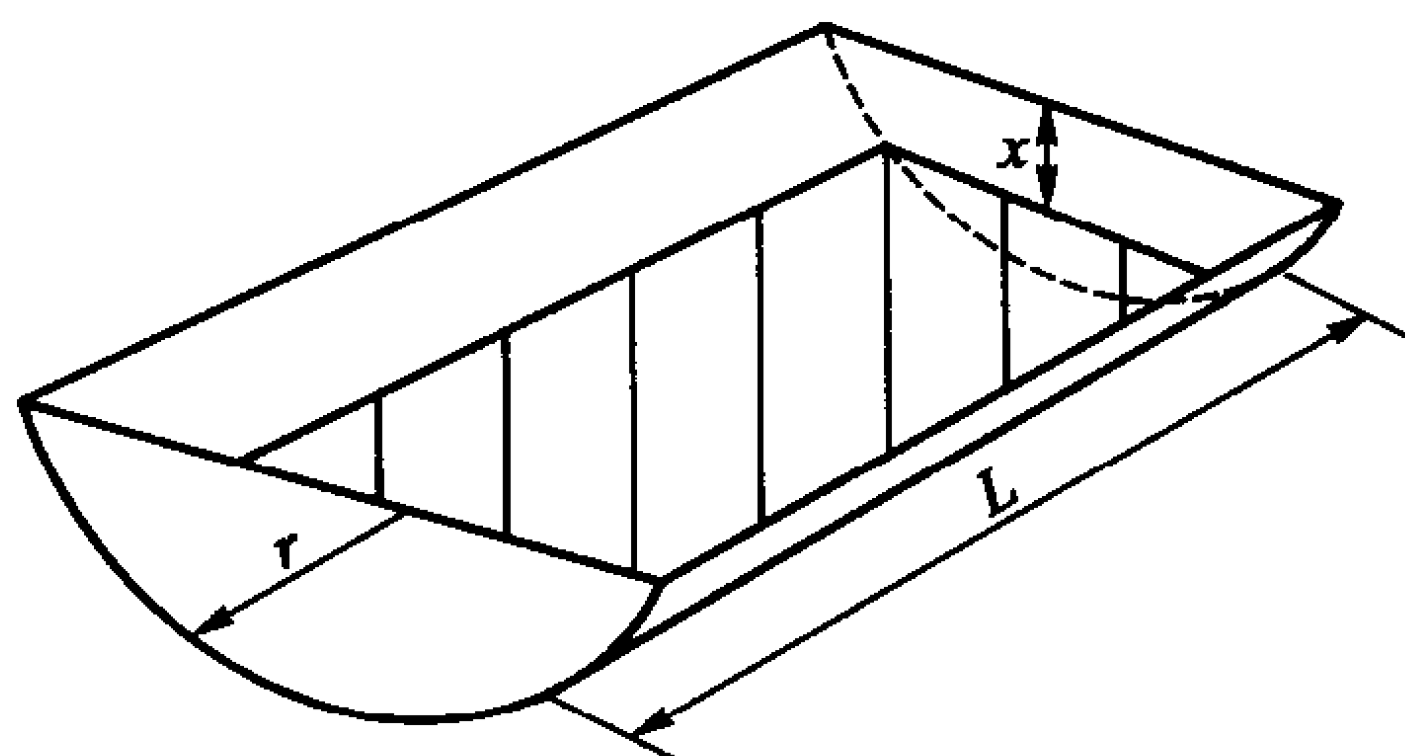


图 2-30

11. 某地区现有人口二百万，十年前为一百万，又知平均每年净迁入人口八万，问十年来人口的平均增长率是多少.

12. 炮弹发射视为斜抛运动，已知初速为 200 m/s，问要击中水平距离 360 m、垂直距离 160 m 的目标，当忽略空气阻力时，发射角应多大. 如果只考虑水平方向的阻力，且设阻力与(水平方向)速度成正比，系数为 0.1 s^{-1}，结果又如何.

13. 分别用迭代公式(2.54)和(2.57)求解下列方程组，要求精度 $\varepsilon = 10^{-6}$.

(1) $\begin{cases} x^2 + y^2 - 2x = 3, \\ 4x^2 - y^2 = 1; \end{cases}$ (2) $\begin{cases} 2x^2 - 2y - 4x + 1 = 0, \\ x^2 + 4y^2 = 4. \end{cases}$

14. 分别用割线法、牛顿切线法、抛物线法、加权迭代法和艾特肯加速方法求下列方程的数值解，并且对各种方法比较.

(1) $x^3 = 3x^2 - 2$ 给定区间(0.7, 1.2)内的数值解，精确到 $\varepsilon = 10^{-4}$;

(2) $e^x = 1/x$ 在 0.5 附近的数值解,精确到 $\varepsilon = 10^{-5}$;

(3) $4x^2 = e^x$ 的数值解,精确到 $\varepsilon = 10^{-9}$.

15. 给定迭代公式 $x_{k+1} = \varphi(x_k)$,其中 $\varphi(x) = x - \dfrac{(m-1)x^m + (m+1)\alpha}{(m+1)x^m + (m-1)\alpha}$,$m \geqslant 2$,并且假设 x_0 充分接近 $x^m - \alpha = 0$ 的某个根 x^*,试证 $\{x_k\}$ 至少具有三阶收敛速度.

16. 不用除法运算,如何求 $1/c$ (其中 $c > 1$)的值?

17. 用牛顿切线法求下列各式的值,精确到 $\varepsilon = 10^{-14}$.

(1) $\sqrt[4]{3}$; (2) $\sqrt[5]{7}$; (3) $\sqrt{2}$.

18. 求抛物线 $y = x^2 + 2$ 与曲线 $y = \dfrac{x}{5} - \sin x$ 之间的最小垂直距离处的 x 值,精确 $\varepsilon = 10^{-10}$.

第三章 解线性方程组的直接方法

在科技、工程、医学和经济等各个领域中，经常遇到求解包含 n 个未知数、由 m 个方程构成的线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \cdots\cdots\cdots\cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases} \tag{3.1}$$

的问题. 线性方程组(3.1)的解法一般有两类：

1. 直接法 经过有限次算术运算求出精确解(如果计算过程中没有舍入误差). 但实际计算中由于有舍入误差的存在和影响，只能得到近似解. 本章介绍这类方法中最基本的高斯(Gauss)消元法、列主元消元法、直接三角形分解法及其 MATLAB 程序等等，并对误差分析作简单讨论. 这类方法是解低阶稠密矩阵方程组的有效方法，也可用于求解具有较大型稀疏矩阵方程组.

2. 迭代法 从初始解出发，根据设计好的步骤用逐次求出的近似解逼近精确解的方法. 常用的迭代法有雅可比(Jacobi)迭代法和高斯-塞德尔(Gauss-Seidel)迭代法等. 迭代法具有需要计算机的存储单元较少、程序设计简单、原始系数矩阵在计算中始终不变等优点，但存在收敛性和收敛速度问题(见第四章). 迭代法是求解大型稀疏矩阵方程组，尤其是由微分方程离散后得到的大型方程组的重要方法.

这些方法适用于实际问题中出现的 n 达到几百、几千甚至上万的方程组. 例如，大型的土建结构、机械结构，大型的输电网络、管道网络，简单的分析可以直接归结为线性方程组，复杂一些要用到偏微分方程，求数值解时将转化为 n 非常大的方程组. 在这章中我们要学习线性方程组的直接法，特别是适合用数学软件在计算机上求解的方法.

3.1 方程组的逆矩阵解法及其 MATLAB 程序

3.1.1 逆矩阵、行列式及其 MATLAB 命令

定义 3.1 对于 n 阶矩阵 $\boldsymbol{A}$，如果有一个 n 阶矩阵 $\boldsymbol{B}$，使

$$\boldsymbol{AB} = \boldsymbol{BA} = \boldsymbol{E},$$

则称矩阵 A 是**可逆的**，并把矩阵 B 称为 A 的**逆矩阵**.

A 的逆矩阵记作 A^{-1}，则 $B=A^{-1}$ 且 $AA^{-1}=A^{-1}A=E$.

当矩阵 A 为方阵时，A 的行列式通常表示为 $\det A$ 或 $|A|$. 当 $\det A\neq 0$ 时，A 可逆，且可以用表 3-1 中的 MATLAB 命令求 A 的逆矩阵和行列式.

表 3-1

命　令	功　　能
NA = inv(A)	输入矩阵 A，运行后输出 A 的逆矩阵 A^{-1}；
HA = det(A)	输入矩阵 A，运行后输出 A 的行列式 $\det A$ 的值.

例 3.1.1 求下列矩阵的逆矩阵和行列式

$$(1)\ A=\begin{pmatrix}1 & -5\\ -1 & 4\end{pmatrix};\quad (2)\ B=\begin{pmatrix}1 & 2 & -3\\ 2 & 0 & 4\\ 0 & -1 & 5\end{pmatrix};\quad (3)\ C=\begin{pmatrix}4 & 2 & 3\\ 0 & -1 & 5\\ 2 & 1 & d\end{pmatrix}.$$

解 在 MATLAB 工作窗口输入程序

```
>> A=[1 -5;-1 4]; NA=inv(A), B=[1 2 -3;2 0 4; 0 -1 5]; NB=
inv(B),
syms d
C=[4 2 3; 0 -1 5; 2 1 d]; NC=inv(C),
HA=det(A), HB =det(B), HC =det(C)
```

运行后分别输出 A,B,C 的逆矩阵和行列式为

```
NA =      -4      -5
          -1      -1
NB = -0.40000000000000    0.70000000000000   -0.80000000000000
      1.00000000000000   -0.50000000000000    1.00000000000000
      0.20000000000000   -0.10000000000000    0.40000000000000
NC =[ 1/2*(d+5)/(2*d-3),  1/2,   -13/2/(2*d-3)]
    [          -5/(2*d-3),   -1,       10/(2*d-3)]
    [          -1/(2*d-3),    0,        2/(2*d-3)]
HA = -1,HB = -10,HC - -4*d+6
```

3.1.2 方程组的逆矩阵解法及其 MATLAB 程序

在 MATLAB 中引进了矩阵除法的概念，它包括左除和右除. 设两矩阵 A,B 为 n 阶方阵，且 A,B 皆可逆，则矩阵方程 $AX=C,YB=D,AZB=F$ 的解都可以用表 3-2 中的命令求出.

表 3-2

命　令	相同功能的命令	功　能
X=A\C	X= inv(A)*C	输入矩阵 A 和 C,运行后输出矩阵方程 $AX=C$ 的解 $X=A^{-1}C$
Y=D/B	Y=D*inv(B)	输入矩阵 B 和 D,运行后输出矩阵方程 $YB=D$ 的解 $Y=DB^{-1}$
Z=A\F/B	Z= inv(A)*F* inv(B)	输入矩阵 A,B 和 F,运行后输出矩阵方程 $AZB=F$ 的解 $Z=A^{-1}FB^{-1}$

例 3.1.2 解下列矩阵方程

(1) $\begin{pmatrix}1 & -5\\ -1 & 4\end{pmatrix}X=\begin{pmatrix}3 & 2\\ 1 & 4\end{pmatrix}$;(2) $Y\begin{pmatrix}1 & -1 & 1\\ 1 & 1 & 0\\ 2 & 1 & 1\end{pmatrix}=\begin{pmatrix}1 & 2 & -3\\ 2 & 0 & 4\\ 0 & -1 & 5\end{pmatrix}$;

(3) $\begin{pmatrix}1 & -1 & 1\\ 1 & 1 & 0\\ 2 & 1 & 1\end{pmatrix}Z\begin{pmatrix}1 & -1 & 1\\ 1 & 1 & 0\\ 3 & 2 & 1\end{pmatrix}=\begin{pmatrix}4 & 2 & 3\\ 0 & -1 & 5\\ 2 & 1 & 1\end{pmatrix}$.

解 下面用两种方法求矩阵方程的解.

方法 1 在 MATLAB 工作窗口输入程序

```
>> A1 =[1 -5; -1 4];C =[3 2; 1 4]; X =A1 \C,
B1 =[1 -1 1;1 1 0;2 1 1]; D =[1 2 -3;2 0 4; 0 -1 5];
Y =D /B1, A2 =[1 -1 1;1 1 0;2 1 1];
B2 =[1 -1 1;1 1 0;3 2 1];F =[4 2 3; 0 -1 5; 2 1 1];Z =A2 \F /B2
```

运行后输出矩阵方程(1)、(2)、(3)的解分别为

```
X =   -17   -28
       -4    -6
Y =

        2     9   -5
       -2    -8    6
       -4   -14    9
Z =
  -8.99999999999999  -51.99999999999999   20.99999999999999
   4.99999999999999   28.99999999999999  -12.00000000000000
  12.99999999999999   73.99999999999999  -28.99999999999999
```

方法 2 在 MATLAB 工作窗口输入程序

```
>>  A1 =[1 -5; -1 4];C =[3 2; 1 4]; X = inv(A1) *C,
```

```
B1 =[1 -1 1;1 1 0;2 1 1]; D =[1 2 -3;2 0 4; 0 -1 5];
Y = D * inv(B1), A2 =[1 -1 1;1 1 0;2 1 1];
B2 =[1 -1 1;1 1 0;3 2 1]; F =[4 2 3; 0 -1 5; 2 1 1];
Z = inv(A2) * F * inv(B2)
```

运行后输出矩阵方程(1)、(2)的解与方法 1 相同,矩阵方程(3)的解

```
Z = -9.00000000000000  -52.00000000000000   21.00000000000000
     5.00000000000000   29.00000000000000  -12.00000000000000
    13.00000000000000   74.00000000000001  -29.00000000000001
```

在 MATLAB 工作窗口输入程序

```
>> (A2 \ F) /B2 - inv(A2) * F * inv(B2)
```

运行后输出矩阵方程(3)的两种解法的解的差为

```
ans =   1.0e-013 *
     0.07105427357601    0.14210854715202   -0.10658141036402
    -0.07105427357601   -0.07105427357601    0.03552713678801
    -0.17763568394003   -0.28421709430404    0.14210854715202
```

例 3.1.3 利用逆矩阵解下列线性方程组

(1) $\begin{cases} x_1 + 2x_2 + 3x_3 = 1, \\ 2x_1 + 2x_2 + 5x_3 = 2, \\ 3x_1 + 5x_2 + x_3 = 3; \end{cases}$ (2) $\begin{cases} x_1 - x_2 - x_3 = 2, \\ 2x_1 - x_2 - 3x_3 = 1, \\ 3x_1 + 2x_2 - 5x_3 = 0. \end{cases}$

解 (1) 方程组可表示为 $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 2 & 5 \\ 3 & 5 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$.

在 MATLAB 工作窗口输入程序

```
>> A =[1 2 3;2 2 5;3 5 1];C =[1;2;3]; X = A \C
```

或

```
>> A =[1 2 3;2 2 5;3 5 1];C =[1;2;3]; X = inv(A) * C
```

运行后输出线性方程组(1)的解为 $x_1 = 1, x_2 = 0, x_3 = 0$.

(2) 同理,在 MATLAB 工作窗口输入程序

```
>> A =[1 -1 -1;2 -1 -3;3 2 -5];C =[2;1;0]; X = A \C
```

运行后输出线性方程组(2)的解为 $x_1 = 5, x_2 = 0, x_3 = 3$.

3.1.3 线性方程组有解的判定条件及其 MATLAB 程序

定理 3.1 n 元线性方程组 $\boldsymbol{A}_{m\times n}\boldsymbol{X} = \boldsymbol{b}$ 有解的充分必要条件是系数矩阵的秩等于增广矩阵 $\boldsymbol{B} = (\boldsymbol{A} \vdots \boldsymbol{b})$ 的秩,即 $R(\boldsymbol{A}) = R(\boldsymbol{B})$.

当 $R(\boldsymbol{A}) = R(\boldsymbol{B}) < n$ 时,n 元非齐次线性方程组 $\boldsymbol{A}_{m\times n}\boldsymbol{X} = \boldsymbol{b}$ 有无穷多解;

当 $R(\boldsymbol{A}) = R(\boldsymbol{B}) = n$ 时,n 元非齐次线性方程组 $\boldsymbol{A}_{m\times n}\boldsymbol{X} = \boldsymbol{b}$ 有唯一解;

当 $R(\boldsymbol{A})\neq R(\boldsymbol{B})$ 时，n 元非齐次线性方程组 $\boldsymbol{A}_{m\times n}\boldsymbol{X}=\boldsymbol{b}$ 无解.

解非齐次线性方程组，将其增广矩阵化成行阶梯形矩阵，便可判断其是否有解．若有解，化成行最简形矩阵，便可写出其通解.

判定线性方程组 $\boldsymbol{A}_{m\times n}\boldsymbol{X}=\boldsymbol{b}$ 是否有解的 MATLAB 程序

输入的量：系数矩阵 $\boldsymbol{A}$ 和常数向量 $\boldsymbol{b}$；

输出的量：RA 和 RB 分别是系数矩阵 $\boldsymbol{A}$ 和增广矩阵 $\boldsymbol{B}$ 的秩，n 是方程组中未知量的个数，另外还输出有关方程组解的信息.

说明 (1) 求矩阵 $\boldsymbol{A}$ 的秩用 MATLAB 命令：rank(A).

(2) 如果有唯一解，则用表 3－2 方法求解；如果有无穷多解，则将其增广矩阵化成行最简形矩阵，便可写出其通解.

根据线性方程组有解的判定条件，现编写了判定线性方程组是否有解的 MATLAB 程序

```
function [RA,RB,n] = jiepb(A,b)
    B = [A b];n = length(b); RA = rank(A);
    RB = rank(B);zhica = RB - RA;
if zhica >0,
    disp('请注意:因为 RA ~ = RB,所以此方程组无解.')
  return,warning off MATLAB;return_outside_of_loop
end
if RA = = RB
  if RA = = n
    disp('请注意:因为 RA = RB = n,所以此方程组有唯一解.')
  else
    disp('请注意:因为 RA = RB < n,所以此方程组有无穷多解.')
  end
end
```

例 3.1.4 判断下列线性方程组解的情况．如果有唯一解，则用表 3－2 方法求解.

(1) $\begin{cases}2x_1+3x_2-x_3+5x_4=0,\\3x_1+x_2+2x_3-7x_4=0,\\4x_1+x_2-3x_3+6x_4=0,\\x_1-2x_2+4x_3-7x_4=0;\end{cases}$ (2) $\begin{cases}3x_1+4x_2-5x_3+7x_4=0,\\2x_1-3x_2+3x_3-2x_4=0,\\4x_1+11x_2-13x_3+16x_4=0,\\7x_1-2x_2+x_3+3x_4=0;\end{cases}$

(3) $\begin{cases}4x_1+2x_2-x_3=2,\\3x_1-x_2+2x_3=10,\\11x_1+3x_2=8;\end{cases}$ (4) $\begin{cases}2x+y-z+w=1,\\4x+2y-2z+w=2,\\2x+y-z-w=1.\end{cases}$

解 (1) **步骤1** 保存名为 jiepb.m 的 M 文件;

步骤2 在 MATLAB 工作窗口输入程序

```
>> A=[2 3 -1 5;3 1 2 -7;4 1 -3 6;1 -2 4 -7];
   b=[ 0; 0; 0; 0]; [RA,RB,n]=jiepb(A,b)
```

步骤3 运行后输出结果为

```
请注意:因为 RA=RB=n,所以此方程组有唯一解.
RA = 4,RB =4,n =4
```

步骤4 在 MATLAB 工作窗口输入

```
>>X=A\b,
```

运行后输出结果

```
X =(0 0 0 0)'.
```

(2) 在 MATLAB 工作窗口输入程序

```
>> A=[3 4 -5 7;2 -3 3 -2;4 11 -13 16;7 -2 1 3];b=[ 0; 0; 0; 0];
   [RA,RB,n]=jiepb(A,b)
```

运行后输出结果

```
请注意:因为 RA=RB<n,所以此方程组有无穷多解.
RA =2,RB =2,n =4
```

(3) 在 MATLAB 工作窗口输入程序

```
>> A=[4 2 -1;3 -1 2;11 3 0]; b=[2;10;8]; [RA,RB,n]=jiepb(A,b)
```

运行后输出结果

```
请注意:因为 RA~=RB,所以此方程组无解.
RA =2,RB =3,n =3
```

(4) 在 MATLAB 工作窗口输入程序

```
>> A=[2 1 -1 1;4 2 -2 1;2 1 -1 -1];
b=[1; 2; 1]; [RA,RB,n]=jiepb(A,b)
```

运行后输出结果

```
请注意:因为 RA=RB<n,所以此方程组有无穷多解.
RA =2,RB =2,n =3
```

例 3.1.5 求解齐次线性方程组
$$\begin{cases} x_1+2x_2+2x_3+x_4=0, \\ 2x_1+x_2-2x_3-2x_4=0, \\ x_1-x_2-4x_3-3x_4=0. \end{cases}$$

解 对系数矩阵 $\boldsymbol{A}$ 施行初等行变换

$$\boldsymbol{A}=\begin{pmatrix} 1 & 2 & 2 & 1 \\ 2 & 1 & -2 & -2 \\ 1 & -1 & -4 & -3 \end{pmatrix} \to \begin{pmatrix} 1 & 2 & 2 & 1 \\ 0 & -3 & -6 & -4 \\ 0 & -3 & -6 & -4 \end{pmatrix}$$

$$\rightarrow\begin{pmatrix}1 & 2 & 2 & 1\\0 & 1 & 2 & \frac{4}{3}\\0 & 0 & 0 & 0\end{pmatrix}\rightarrow\begin{pmatrix}1 & 0 & -2 & -\frac{5}{3}\\0 & 1 & 2 & \frac{4}{3}\\0 & 0 & 0 & 0\end{pmatrix},$$

得与原方程组同解的方程组 $\begin{cases}x_1 = 2x_3 + \frac{5}{3}x_4,\\x_2 = -2x_3 - \frac{4}{3}x_4.\end{cases}$ 令 $x_3 = c_1, x_4 = c_2$,

得
$$\begin{cases}x_1 = 2c_1 + \frac{5}{3}c_2,\\x_2 = -2c_1 - \frac{4}{3}c_2,\\x_3 = c_1,\\x_4 = c_2,\end{cases}$$

因此,
$$\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix} = c_1\begin{pmatrix}2\\-2\\1\\0\end{pmatrix} + c_2\begin{pmatrix}\frac{5}{3}\\-\frac{4}{3}\\0\\1\end{pmatrix}.$$

例 3.1.6 用初等行变换解例 3.1.4 中非齐次线性方程组(4).

解 对方程组的增广矩阵施行初等行变换,化为行最简型矩阵为

$$\boldsymbol{B} = \begin{pmatrix}2 & 1 & -1 & 1 & 1\\4 & 2 & -2 & 1 & 2\\2 & 1 & -1 & -1 & 1\end{pmatrix}\xrightarrow{\text{初等行变换}}\begin{pmatrix}2 & 1 & -1 & 0 & 1\\0 & 0 & 0 & 1 & 0\\0 & 0 & 0 & 0 & 0\end{pmatrix},$$

得 $\begin{cases}x = x\\y = y\\z = 2x + y - 1\\w = 0\end{cases}$,即 $\begin{pmatrix}x\\y\\z\\w\end{pmatrix} = k_1\begin{pmatrix}1\\0\\2\\0\end{pmatrix} + k_2\begin{pmatrix}0\\1\\1\\0\end{pmatrix} + \begin{pmatrix}0\\0\\-1\\0\end{pmatrix}$, 其中 k_1, k_2 为任意常数.



习 题 3.1

1. 求下列方阵的秩:

(1) $\begin{pmatrix}1 & 0 & 2 & -1\\2 & 0 & 3 & 1\\3 & 0 & 4 & -3\end{pmatrix}$; (2) $\begin{pmatrix}0 & 2 & -3 & 1\\0 & 3 & -4 & 3\\0 & 4 & -7 & -1\end{pmatrix}$;

(3) $\begin{pmatrix} 1 & -1 & 3 & -4 & 3 \\ 3 & -3 & 5 & -4 & 1 \\ 2 & -2 & 3 & -2 & 0 \\ 3 & -3 & 4 & -2 & -1 \end{pmatrix}$; (4) $\begin{pmatrix} 2 & 3 & 1 & -3 & -7 \\ 1 & 2 & 0 & -2 & -4 \\ 3 & -2 & 8 & 3 & 0 \\ 2 & -3 & 7 & 4 & 3 \end{pmatrix}$.

2. 求下列方阵的逆矩阵：

(1) $\begin{pmatrix} 3 & 2 & 1 \\ 3 & 1 & 5 \\ 3 & 2 & 3 \end{pmatrix}$; (2) $\begin{pmatrix} 3 & -2 & 0 & -1 \\ 0 & 2 & 2 & 1 \\ 1 & -2 & -3 & -2 \\ 0 & 1 & 2 & 1 \end{pmatrix}$.

3. 解下列矩阵方程：

(1) 设 $\boldsymbol{A} = \begin{pmatrix} 4 & 1 & -2 \\ 2 & 2 & 1 \\ 3 & 1 & -1 \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} 1 & -3 \\ 2 & 2 \\ 3 & -1 \end{pmatrix}$，求 $\boldsymbol{X}$ 使 $\boldsymbol{AX} = \boldsymbol{B}$；

(2) 设 $\boldsymbol{A} = \begin{pmatrix} 0 & 2 & 1 \\ 2 & -1 & 3 \\ -3 & 3 & -4 \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & -3 & 1 \end{pmatrix}$，求 $\boldsymbol{X}$ 使 $\boldsymbol{XA} = \boldsymbol{B}$；

(3) $\boldsymbol{A} = \begin{pmatrix} 11 & -1 & 1 \\ 1 & 1 & 0 \\ 2 & 1 & 1 \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} 3 & -1 & 1 \\ 1 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix}, \boldsymbol{C} = \begin{pmatrix} 4 & 2 & 3 \\ 0 & -1 & 5 \\ 2 & 1 & 1 \end{pmatrix}$，求 $\boldsymbol{X}$ 使 $\boldsymbol{AXB} = \boldsymbol{C}$.

4. 求下列行列式：

(1) $\begin{vmatrix} 4 & 1 & 2 & 4 \\ 1 & 2 & 0 & 2 \\ 10 & 5 & 2 & 0 \\ 0 & 1 & 1 & 7 \end{vmatrix}$; (2) $\begin{vmatrix} 2 & 1 & 4 & 1 \\ 3 & -1 & 2 & 1 \\ 1 & 2 & 3 & 2 \\ 5 & 0 & 6 & 2 \end{vmatrix}$;

(3) $\begin{vmatrix} -ab & ac & ae \\ bd & -cd & de \\ bf & cf & -ef \end{vmatrix}$; (4) $\begin{vmatrix} a & 1 & 0 & 0 \\ -1 & b & 1 & 0 \\ 0 & -1 & c & 1 \\ 0 & 0 & -1 & d \end{vmatrix}$.

5. 判断下列线性方程组解的情况，如果有唯一解，则求出解.

(1) $\begin{cases} x_1 + x_2 + x_3 + x_4 = 5, \\ x_1 + 2x_2 - x_3 + 4x_4 = -2, \\ 2x_1 - 3x_2 - x_3 - 5x_4 = -2, \\ 3x_1 + x_2 + 2x_3 + 11x_4 = 0; \end{cases}$ (2) $\begin{cases} 5x_1 + 6x_2 = 1 \\ x_1 + 5x_2 + 6x_3 = 0, \\ x_2 + 5x_3 + 6x_4 = 0, \\ x_3 + 5x_4 + 6x_5 = 0, \\ x_4 + 5x_5 = 1; \end{cases}$

(3) $\begin{cases} x_1 - 2x_2 + 3x_3 - x_4 = 1, \\ 3x_1 - x_2 + 5x_3 - 3x_4 = 2, \\ 2x_1 + x_2 + 2x_3 - 2x_4 = 3; \end{cases}$ (4) $\begin{cases} x_1 + 2x_2 + 2x_3 + x_4 = 0, \\ 2x_1 + x_2 - 2x_3 - 2x_4 = 0, \\ x_1 - x_2 - 4x_3 - 3x_4 = 0. \end{cases}$

3.2 三角形方程组的解法及其 MATLAB 程序

用初等行变换解线性方程组是解线性方程组的一般方法. 从例 3.1.6 可见,当方程的个数和未知量的个数很大时,用初等行变换解线性方程组的工作量相当的大,如果在计算过程中某一步发生错误,那么就会得出错误的结果. 人们非常想用计算机的程序解线性方程组,而高斯消元法以及各种变形的高斯消元法使得用初等行变换解线性方程组实现机械化计算成为可能.

因为高斯消元法以及各种变形的高斯消元法都归结为下三角形方程组和上三角形方程组的解法,所以首先介绍三角形方程组的解法. 下面介绍解上三角形线性方程组回代法,而将下三角形线性方程组回代法留给读者完成.

3.2.1 三角形方程组的解法

设上三角形线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1,n-1}x_{n-1} + a_{1n}x_n = b_1, \\ \qquad a_{22}x_2 + \cdots + a_{2,n-1}x_{n-1} + a_{2n}x_n = b_2, \\ \qquad \cdots\cdots\cdots\cdots \\ \qquad\qquad a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1}, \\ \qquad\qquad\qquad a_{n,n}x_n = b_n \end{cases} \tag{3.2}$$

的系数矩阵、常系数向量和未知数的向量分别为

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n-1} & a_{1n} \\ & a_{22} & \cdots & a_{2n-1} & a_{2n} \\ & & \ddots & \vdots & \vdots \\ & & & a_{n-1,n-1} & a_{n-1,n} \\ & & & & a_{nn} \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix},$$

$$\boldsymbol{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} \text{(其中 } \boldsymbol{A} \text{ 称为上三角形矩阵)},$$

则(3.2)式可表示为矩阵方程

$$\boldsymbol{AX} = \boldsymbol{b}. \tag{3.3}$$

定理 3.2 如果上三角形线性方程组(3.2)的系数矩阵 $\boldsymbol{A}$ 中的主对角线上的所有元

$$a_{kk}\neq 0,\quad k=1,2,\cdots,n, \tag{3.4}$$

则方程组(3.2)有唯一解.

证明 因为 $a_{kk}\neq 0,\quad k=1,2,\cdots,n,$,所以从(3.2)式的最后一个方程解出

$$x_n=\frac{b_n}{a_{nn}},$$

现在 x_n 已知,将它代入它上面的一个方程解出

$$x_{n-1}=\frac{b_{n-1}-a_{n-1,n}x_n}{a_{n-1,n-1}},$$

现在 x_n 和 x_{n-1} 已知,将它们代入它们上面的一个方程后,解出

$$x_{n-2}=\frac{b_{n-2}-a_{n-2,n-1}x_{n-1}-a_{n-2,n}x_n}{a_{n-2,n-2}}.$$

当 $x_n,\cdots,x_{k+1}$ 都求出之后,则可得到一般步骤为

$$\begin{cases} x_k=\dfrac{b_k-\sum\limits_{j=k+1}^{n}a_{kj}x_j}{a_{kk}}, & k=n-1,n-2,\cdots,1. \\ x_n=\dfrac{b_n}{a_{nn}}, \end{cases} \tag{3.5}$$

并如此进行下去,即可依次将 $x_n,\cdots,x_1$ 全部解出. 根据第 n 个方程可推出 $x_n=\dfrac{b_n}{a_{nn}}$ 是 x_n 的唯一可能值,然后用有限数学归纳法可证明 $x_n,\cdots,x_1$ 是唯一的解. 这种方法称**回代法**.

例 3.2.1 用回代法解上三角形线性方程组

$$\begin{cases} 5x_1-x_2+2x_3+3x_4=20, \\ \quad -2x_2+7x_3-4x_4=-7, \\ \quad\quad 6x_3+5x_4=4, \\ \quad\quad\quad 3x_4=6. \end{cases} \tag{3.6}$$

解 因为 $a_{kk}\neq 0,k=1,2,3,4$,所以从(3.6)式的最后一个方程解出 $x_4=2$. 将 x_4 代入三个方程解出 $x_3=-1$. 现在 $x_3=-1$ 和 $x_4=2$ 代入二个方程后,解出

$$x_2=\frac{-7-7\cdot(-1)+4\cdot 2}{-2}=-4.$$

最后求解出第一个方程中的 x_1,可得

$$x_1=\frac{20+1\cdot(-4)-2\cdot(-1)-3\cdot 2}{5}=2.4.$$

定理 3.3 如果 $n\times n$ 阶矩阵 $\boldsymbol{A}$ 是上三角形矩阵或下三角形矩阵,则当

$$\det \boldsymbol{A}=a_{11}a_{22}\cdots a_{nn}=\prod_{k=1}^{n}a_{kk}\neq 0$$

时,$\boldsymbol{A}$ 对应的上三角形线性方程组或下三角形线性方程组有唯一解.

3.2.2 解三角形方程组的 MATLAB 程序

下面编写的 MATLAB 程序是利用回代法解上三角形线性方程组,解下三角形线性方程组的程序类似.

解上三角形线性方程组 $AX=B$ 的 MATLAB 程序

输入的量:系数矩阵 A 和常系数向量 b;

运行后输出的量:系数矩阵 A 和增广矩阵 B 的秩 RA,RB, 方程组中未知量的个数 n 和有关方程组解 X 及其解的信息.

```
function [RA,RB,n,X] = shangsan(A,b)
    B = [A b]; n = length(b); RA = rank(A); RB = rank(B);zhica = RB - RA;
if zhica >0,
    disp('请注意:因为 RA ~ = RB,所以此方程组无解.')
  return,warning off MATLAB:return_outside_of_loop
end
if RA = = RB
    if RA = = n
    disp('请注意:因为 RA = RB = n,所以此方程组有唯一解.')
        X = zeros(n,1); X(n) = b(n)/A(n,n);
      for k = n - 1: - 1:1
          X(k) = (b(k) - sum(A(k,k + 1:n) * X(k + 1:n)))/A(k,k);
      end
    else
      disp('请注意:因为 RA = RB < n,所以此方程组有无穷多解.')
    end
end
```

例 3.2.2 用解上三角形线性方程组的 MATLAB 程序解例 3.2.1 中的方程组.

解 (1) 分别写出方程组的系数矩阵和常系数向量

$$A=\begin{pmatrix}5 & -1 & 2 & 3\\ 0 & -2 & 7 & -4\\ 0 & 0 & 6 & 5\\ 0 & 0 & 0 & 3\end{pmatrix},b=\begin{pmatrix}20\\ -7\\ 4\\ 6\end{pmatrix}.$$

(2) 保存名为 shangsan.m 的 M 文件;

(3) 在 MATLAB 工作窗口输入程序

```
>>A = [5  -1 2 3;0  -2 7  -4;0 0 6 5;0 0 0 3];
  b = [20;  -7; 4;6]; [RA,RB,n,X] = shangsan(A,b)
```

(4) 运行后输出结果

请注意:因为 RA = RB = n,所以此方程组有唯一解.

```
RA = 4,RB = 4,n = 4,X =[2.4   -4.0   -1.0   2.0]'
```



习题 3.2

1. 用回代法解上三角形线性方程组

(1) $\begin{cases} 4x_1 + 8x_2 + 4x_3 = 8, \\ 3x_2 + 3x_3 - 3x_4 = -6, \\ x_3 + x_4 = 3, \\ 3x_4 = 6; \end{cases}$ (2) $\begin{cases} x_1 + 2x_2 - x_4 = 9, \\ -x_2 - x_3 + 2x_4 = -9, \\ -2x_3 + 3x_4 = -10, \\ 3x_4 = -6. \end{cases}$

2. 用回代法解下三角形线性方程组

(1) $\begin{cases} -6x_1 = -24, \\ -2x_1 + 3x_2 = 6, \\ 4x_1 - 2x_2 + 2x_3 = 10, \\ x_1 + 2x_2 + 4x_3 + x_4 = 21; \end{cases}$ (2) $\begin{cases} -2x_1 = -42, \\ 2x_1 + x_2 = 52, \\ 3x_1 + x_2 + x_3 = 79, \\ 4x_1 + x_2 + 2x_3 + x_4 = 82. \end{cases}$

3.3 高斯(Gauss)消元法和列主元消元法及其 MATLAB 程序

3.3.1 高斯消元法及其 MATLAB 程序

定义 3.2 在(3.1)中设 $m=n, a_{11}\neq 0$ 将第 1 行乘以 $(-a_{k1}/a_{11})$,加入第 k 行($k=2,...,n$),得

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)}, \\ \qquad a_{22}^{(2)}x_2 + \cdots + a_{2n}^{(2)}x_n = b_2^{(2)}, \\ \qquad \cdots\cdots\cdots\cdots \\ \qquad a_{n2}^{(2)}x_2 + \cdots + a_{nn}^{(2)}x_n = b_n^{(2)}, \end{cases} \tag{3.7}$$

(其中 $a_{1k}^{(1)}=a_{1k}, b_1^{(1)}=b_1$). 再设 $a_{22}^{(2)}\neq 0$,将(3.7)式的第 2 行乘以 $(-a_{k2}^{(2)}/a_{22}^{(2)})$,加入第 k 行($k=3,\cdots,n$),并如此进行下去,最终得到上三角形线性方程组

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1,n-1}^{(1)}x_{n-1} + a_{1n}^{(1)}x_n = b_1^{(1)}, \\ \qquad a_{22}^{(2)}x_2 + \cdots + a_{2,n-1}^{(2)}x_{n-1} + a_{2n}^{(2)}x_n = b_2^{(2)}, \\ \qquad \cdots\cdots\cdots\cdots \\ \qquad\qquad a_{n-1,n-1}^{(n-1)}x_{n-1} + a_{n-1,n}^{(n-1)}x_n = b_{n-1}^{(n-1)}, \\ \qquad\qquad\qquad a_{n,n}^{(n)}x_n = b_n^{(n)}. \end{cases} \tag{3.8}$$

用上节介绍的回代法解上三角形线性方程组(3.8),即可依次将 $x_n,\cdots,x_1$ 全部解出.这样,在 $a_{kk}^{(k)}\neq 0\ (k=1,2,\cdots,n)$ 的假设下,由上而下的消元和由下而上的回代,就构成了方程组的消元法,称**高斯消元法**.

消元法的步骤可用矩阵、向量符号描述.

若记 $\boldsymbol{A}=(a_{ij})_{n\times n}$,$\boldsymbol{X}=(x_1,\cdots,x_n)^{\mathrm{T}}$,$\boldsymbol{b}=(b_1,\cdots,b_n)^{\mathrm{T}}$,当 $m=n$ 时,(3.1)式可表示为

$$\boldsymbol{AX}=\boldsymbol{b}. \tag{3.9}$$

定理 3.4 如果 $n\times n$ 阶矩阵 $\boldsymbol{A}$ 的秩 $R(\boldsymbol{A})=n$,那么存在 $n\times n$ 阶上三角形矩阵 $\boldsymbol{U}=(u_{kk})$ 和 $n\times n$ 阶单位下三角形矩阵 $\boldsymbol{M}$(对角元为 1,上三角元为 0),并且 $u_{kk}\neq 0\ (k=1,2,\cdots,n)$ 使得 $\boldsymbol{AX}=\boldsymbol{b}$ 有唯一解

$$\boldsymbol{x}=\boldsymbol{U}^{-1}\boldsymbol{Mb}.$$

证明 由(3.1)化为(3.7)式的过程相当于用**单位下三角形矩阵**(对角元为 1,上三角元都为 0),

$$\boldsymbol{M}_1=\begin{pmatrix} 1 & & & \\ -a_{21}^{(1)}/a_{11}^{(1)} & 1 & & \\ \vdots & & \ddots & \\ -a_{n1}^{(1)}/a_{11}^{(1)} & \cdots & & 1 \end{pmatrix}.$$

左乘(3.9)式,即 $\boldsymbol{M}_1\boldsymbol{Ax}=\boldsymbol{M}_1\boldsymbol{b}$ 再依次分别用单位下三角形矩阵 $\boldsymbol{M}_2,\cdots,\boldsymbol{M}_{n-1}$ 左乘 $\boldsymbol{M}_1\boldsymbol{Ax}=\boldsymbol{M}_1\boldsymbol{b}$ 两端,最终将(3.9)式化为(3.8)式,即

$$\boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{Ax}=\boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b}. \tag{3.10}$$

记 $\boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_2\boldsymbol{M}_1=\boldsymbol{M}$,则 $\boldsymbol{M}$ 仍为单位下三角形矩阵,而按照(3.8)式的上三角形状,$\boldsymbol{MA}$ 已化为**上三角形矩阵**(下三角元为 0),记作 $\boldsymbol{U}$,其对角元为 $u_{kk}=a_{kk}^{(k)}\neq 0\ (k=1,2,\cdots,n)$.于是(3.8)式可写成

$$\boldsymbol{Ux}=\boldsymbol{Mb}. \tag{3.11}$$

(3.11)式是消元法最终得到的(3.8)式的矩阵表达形式.因为 $\boldsymbol{U}$ 可逆,所以方程组(3.11),即方程组(3.9)的解为

$$\boldsymbol{x}=\boldsymbol{U}^{-1}\boldsymbol{Mb}. \tag{3.12}$$

这样,求解(3.8)式的主要过程相当于用单位下三角形矩阵左乘 $\boldsymbol{A}$,将 $\boldsymbol{A}$ 化为上三角形矩阵的过程,而且不难证明,消元中关于 $a_{kk}^{(k)}\neq 0\ (k=1,2,\cdots,n)$ 的

假设,等价于 A 的各阶顺序主子式

$$D_k=\begin{vmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kk} \end{vmatrix}\neq 0\ (k=1,2,\cdots,n).$$

例 3.3.1 用高斯消元法和 MATLAB 程序求解非齐次线性方程组,并且用逆矩阵解方程组的方法验证.

$$\begin{cases} \quad -x_2 -x_3 +x_4 =0, \\ x_1 -x_2 +x_3 -3x_4 =1, \\ 2x_1 -2x_2 -4x_3 +6x_4 = -1, \\ x_1 -2x_2 -4x_3 +x_4 = -1. \end{cases} \tag{3.13}$$

解 步骤 1 将非齐次线性方程组(3.13)的系数保存在增广矩阵 B 中,进行初等行变换,$a_{ij}^{(1)}$ 的上标表示第一次保存在位置(i,j)中的元

$$B=\begin{pmatrix} 0 & -1 & -1 & 1 & 0 \\ 1 & -1 & 1 & -3 & 1 \\ 2 & -2 & -4 & 6 & -1 \\ 1 & -2 & -4 & 1 & -1 \end{pmatrix}.$$

步骤 2 因为 $a_{11}^{(1)}=0$,所以交换第 1 行和第 2 行,然后第 1 行乘以(-2)加到第 3 行,第 1 行乘以(-1)加到第 4 行,即

$$B=\begin{pmatrix} 0 & -1 & -1 & 1 & 0 \\ 1 & -1 & 1 & -3 & 1 \\ 2 & -2 & -4 & 6 & -1 \\ 1 & -2 & -4 & 1 & -1 \end{pmatrix}\xrightarrow{r_1\leftrightarrow r_2}\begin{pmatrix} 1 & -1 & 1 & -3 & 1 \\ 0 & -1 & -1 & 1 & 0 \\ 2 & -2 & -4 & 6 & -1 \\ 1 & -2 & -4 & 1 & -1 \end{pmatrix}$$

$$\xrightarrow[r_4-r_1]{r_3+(-2)r_1}\begin{pmatrix} 1 & -1 & 1 & -3 & 1 \\ 0 & -1 & -1 & 1 & 0 \\ 0 & 0 & -6 & 12 & -3 \\ 0 & -1 & -5 & 4 & -2 \end{pmatrix}.$$

步骤 1 和步骤 2 的计算过程用 MATLAB 程序表述为

```
>> A=[0 -1 -1 1;1 -1 1 -3;2 -2 -4 6;1 -2 -4 1]; b=[0;1;-1;-1];
   B=[A b]; n=length(b); X=zeros(n,1);C=zeros(1,n+1);
   if  (B(1,1)==0)&(B(2,1)~=0)
      C=B(1,:); B(1,:)=B(2,:); B(2,:)=C;
      for r=2:n
      mr1 = B(r,1)/B(1,1); B(r,1)=0; %m_r1 = a_r1^(2)/a_11^(2);a_r1^(3) =0
     for c=2:n+1
      B(r,c) = B(r,c) -mr1 * B(1,c); %a_rc^(3) =a_rc^(2) -m_r1 * a_1c^(2);
```

```
        end
      end
  end
```

步骤 3 将第 2 行乘以(-1)加到第 4 行,得

$$\begin{pmatrix} 1 & -1 & 1 & -3 & 1 \\ 0 & -1 & -1 & 1 & 0 \\ 0 & 0 & -6 & 12 & -3 \\ 0 & 0 & -4 & 3 & -2 \end{pmatrix}.$$

步骤 3 的计算过程用 MATLAB 程序表述为

```
for r =3:n
        mr1 = B(r,1)/B(1,1); B(r,1) =0; % m_r1 = a_r1^(3)/a_11^(3);a_r1^(4) =0
    for c =3:n +1
        B(r,c) = B(r,c) -mr1 * B(1,c);% a_rc^(4) =a_rc^(3) - m_r1 * a_1c^(3);
    end
end
```

步骤 4 将第 3 行乘以($-\frac{4}{6}$)加到第 4 行,此过程用 MATLAB 程序表述为

```
for r =4:n
    mr1 = B(r,1)/B(1,1); B(r,1) =0; % m_r1 =a_r1^(4)/a_11^(4);a_r1^(5) =0
    for c =4:n +1
        B(r,c) = B(r,c) -mr1 * B(1,c);% a_rc^(5) =a_rc^(4) - m_r1 * a_1c^(4);
    end
end
```

得

$$B_1 = \begin{pmatrix} 1 & -1 & 1 & -3 & 1 \\ 0 & -1 & -1 & 1 & 0 \\ 0 & 0 & -6 & 12 & -3 \\ 0 & 0 & 0 & -5 & 0 \end{pmatrix}.$$

步骤 5 用解上三角形线性方程组的 MATLAB 程序解 B_1 对应的方程组. 在 MATLAB 工作窗口输入程序

```
>> A =[1 -1 1 -3;0 -1 -1 1;0 0 -6 12;0 0 0 -5];
  b =[1;0; -3; 0];[RA,RB,n,X] =shangsan(A,b)
```

运行后输出结果

```
请注意:因为 RA =RB =n,所以此方程组有唯一解.
    RA = 4,RB = 4,n = 4,X =[0 -0.5 0.5 0]'
```

步骤 6 并且用逆矩阵解方程组的方法验证. 在 MATLAB 工作窗口输入程序

```
>> A=[0 -1 -1 1;1 -1 1 -3;2 -2 -4 6;1 -2 -4 1];
   b=[0;1; -1; -1];X=A\b
```

运行后输出结果相同.

下面提供的 MATLAB 程序是利用高斯消元法解线性方程组 $AX=b$ 的 MATLAB 主程序.

用高斯消元法解线性方程组 $AX=b$ 的 MATLAB 主程序

输入的量:系数矩阵 A 和常系数向量 b;

输出的量:系数矩阵 A 和增广矩阵 B 的秩 RA,RB,方程组中未知量的个数 n 和有关方程组解 X 及其解的信息.

```
function [RA,RB,n,X]=gaus(A,b)
  B=[A b]; n=length(b); RA=rank(A);
  RB=rank(B);zhica=RB-RA;
  if zhica>0,
      disp('请注意:因为 RA~=RB,所以此方程组无解.')
      return
  end
  if RA==RB
    if RA==n
        disp('请注意:因为 RA=RB=n,所以此方程组有唯一解.')
      X=zeros(n,1); C=zeros(1,n+1);
      for p= 1:n-1
          for k=p+1:n
            m= B(k,p)/B(p,p);
            B(k,p:n+1)= B(k,p:n+1)-m* B(p,p:n+1);
          end
      end
        b=B(1:n,n+1);A=B(1:n,1:n); X(n)=b(n)/A(n,n);
      for q=n-1:-1:1
        X(q)=(b(q)-sum(A(q,q+1:n)*X(q+1:n)))/A(q,q);
      end
    else
        disp('请注意:因为 RA=RB<n,所以此方程组有无穷多解.')
    end
end
```

例 3.3.2 用高斯消元法和 MATLAB 程序求解下面的非齐次线性方程组,并且用逆矩阵解方程组的方法验证.

$$\begin{cases} x_1 - x_2 + x_3 - 3x_4 = 1, \\ \quad\; - x_2 - x_3 + x_4 = 0, \\ 2x_1 - 2x_2 - 4x_3 + 6x_4 = -1, \\ x_1 - 2x_2 - 4x_3 + x_4 = -1. \end{cases}$$

解 在 MATLAB 工作窗口输入程序

```
>> A=[1 -1 1 -3; 0 -1 -1 1;2 -2 -4 6;1 -2 -4 1];
   b=[1;0; -1; -1]; [RA,RB,n,X] =gaus (A,b)
```

运行后输出结果

```
请注意:因为 RA = RB = n,所以此方程组有唯一解.
RA =
     4                          X =
RB =                                    0
     4                            -0.5000
n =                                0.5000
     4                                  0
```

3.3.2 列主元消元法及其 MATLAB 程序

从例 3.3.1 可见,用高斯消元法解线性方程组时,$a_{kk}^{(k)} \neq 0 \ (k=1,2,\cdots,n)$是十分重要的条件.实际上,在用消元法解方程组的过程中,即使 $a_{kk}^{(k)} \neq 0 \ (k=1,2,\cdots,n)$,但是其绝对值$|a_{kk}^{(k)}|(k=1,2,\cdots,n)$很小时,用它作除数会导致舍入误差的增加很大,所以在进行到第 k 步时不论 $a_{ik}^{(k)}$是否为 0,都按列选择$|a_{ik}^{(k)}| \ (i=k,\cdots,n)$中最大的一个,称为**列主元**,将列主元所在行与第 k 行交换后再按上面的高斯消元法进行下去,称为**列主元消元法**.

下面提供的 MATLAB 程序是利用列主元消元法解线性方程组.

用列主元消元法解线性方程组 $AX=b$ 的 MATLAB 程序

输入的量:系数矩阵 A 和常系数向量 b;

输出的量:系数矩阵 A 和增广矩阵 B 的秩 RA,RB,方程组中未知量的个数 n 和有关方程组解 X 及其解的信息.

```
function [RA,RB,n,X] =liezhu(A,b)
  B=[A b]; n=length(b); RA=rank(A);
  RB=rank(B);zhica=RB-RA;
  if zhica>0,
    disp('请注意:因为 RA~ =RB,所以此方程组无解.')
```

```
    return
    warning off MATLAB:return_outside_of_loop
  end
  if RA = =RB
    if RA = =n
          disp('请注意:因为 RA =RB =n,所以此方程组有唯一解.')
        X =zeros(n,1); C =zeros(1,n +1);
        for p = 1:n -1
          [Y,j] =max(abs(B(p:n,p))); C =B(p,:);
          B(p,:) = B(j +p -1,:); B(j +p -1,:) =C;
          for k =p +1:n
              m = B(k,p)/B(p,p);
              B(k,p:n +1) = B(k,p:n +1) -m* B(p,p:n +1);
          end
        end
              b =B(1:n,n +1);A =B(1:n,1:n); X(n) -b(n)/A(n,n);
        for q =n -1: -1:1
              X(q) =(b(q) -sum(A(q,q +1:n) * X(q +1:n)))/A(q,q);
        end
      else
          disp('请注意:因为 RA =RB <n,所以此方程组有无穷多解.')
  end
end
```

例 3.3.3 用列主元消元法解线性方程组的 MATLAB 程序解例 3.3.1 的方程组.

解 在 MATLAB 工作窗口输入程序

```
>> A =[0 -1 -1 1;1 -1 1 -3;2 -2 -4 6;1 -2 -4 1];
   b =[0;1; -1; -1]; [RA,RB,n,X] =liezhu(A,b)
```

运行后输出结果

```
请注意:因为 RA =RB =n,所以此方程组有唯一解.
RA = 4,RB = 4,n = 4,X =[0 -0.5 0.5 0]'
```

例 3.3.4 设抛物线 $y = a + bx + cx^2$ 经过点(0.1,6),(-3,5),(0.07,1),求此抛物线方程.

解 将点(0.1,6),(-3,5),(0.07,1)分别代入抛物线方程 $y = a + bx + cx^2$ 中,

得
$$\begin{cases} a + 0.1b + 0.01c = 6, \\ a - 3b + 9c = 5, \\ a + 0.07b + 0.0049c = 1. \end{cases}$$

在 MATLAB 工作窗口输入程序

```
>> A=[1 0.1 0.01;1 -3 9;1 0.07 0.0049];
b=[6;5;1];
[RA,RB,n,X]=liezhu(A,b)
```

运行后输出结果

```
请注意:因为 RA=RB=n,所以此方程组有唯一解.
RA =3,RB =3 ,n =3 ,X =[ -5298/515  15903/101  15334/283 ]'
```

所求的抛物线方程为 $y=-\frac{5\,298}{515}+\frac{15\,903}{101}x+\frac{15\,334}{283}x^2$.



习 题 3.3

1. 用高斯消元法解下列线性方程组：

(1) $\begin{cases}4x_1+8x_2+4x_3=8,\\ x_1+5x_2+4x_3-3x_4=-4,\\ x_1+4x_2+7x_3+2x_4=10,\\ x_1+3x_2+x_3-7x_4=-4;\end{cases}$ (2) $\begin{cases}x_1+x_2+2x_3+3x_4=1,\\ x_1+3x_2+6x_3+x_4=3,\\ 3x_1-x_2-3x_3+15x_4=3,\\ x_1-5x_2-10x_3+12x_4=5.\end{cases}$

2. 用列主元消元法解下列线性方程组：

(1) $\begin{cases}x_1+20x_2-x_3+0.001x_4=0,\\ 2x_1-5x_2+30x_3-0.1x_4=1,\\ 5x_1+x_2-100x_3-10x_4=0,\\ 2x_1-100x_2-x_3+x_4=0;\end{cases}$ (2) $\begin{cases}x_2+2x_3+0.03x_4=1,\\ 0.1x_1+3x_2+0.6x_3+x_4=0,\\ 3x_1-x_2-3x_3+15x_4=-3,\\ x_1-5x_2-10x_3+12x_4=0.\end{cases}$

3. 设抛物线 $y=a+bx+cx^2$ 经过点(1,6),(3,5),(7,2),求此抛物线方程.

3.4 LU 分解法及其 MATLAB 程序

3.4.1 判断矩阵 LU 分解的充要条件及其 MATLAB 程序

(一)矩阵的 LU 分解

高斯消元法的过程说明,矩阵 $\boldsymbol{A}$ 左乘单位下三角形矩阵 $\boldsymbol{M}$,可化为上三角形矩阵 $\boldsymbol{U}$,即

$$\boldsymbol{MA}=\boldsymbol{U}.$$

因为单位下三角形矩阵的逆仍为单位下三角形矩阵，记 $\boldsymbol{M}^{-1}=\boldsymbol{L}$,则

$$\boldsymbol{A}=\boldsymbol{LU} \tag{3.14}$$

将矩阵 $\boldsymbol{A}$ 分解为一个单位下三角形矩阵 $\boldsymbol{L}$ 和一个上三角形矩阵 $\boldsymbol{U}$ 的积

$A=LU$,这种分解称为**矩阵 LU 分解**. (3.14)式用矩阵的形式可表示为

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix} \tag{3.15}$$

定义 3.3 $m\times n$ 阶矩阵 A 的左上角 $r\times r$ $(1\leqslant r\leqslant \min\{m,n\})$ 阶子矩阵 A_r 的行列式

$$\det A_r = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rr} \end{vmatrix} \tag{3.16}$$

称为 A 的 **r 阶顺序主子式**.

n 阶矩阵 A 能否进行 LU 分解的充要条件,可由如下定理描述.

定理 3.5 n 阶矩阵 A 有唯一的 LU 分解的充要条件是 A 的各阶顺序主子式 $\det A_r$ $(r=1,2,\cdots,n)$ 不为零.

(二) 判断矩阵 A 能否进行 LU 分解的 MATLAB 程序

根据定理 3.5 和(3.16)式编写的 MATLAB 程序如下:

> **判断矩阵 A 能否进行 LU 分解的 MATLAB 程序**
>
> 输入的量:系数矩阵 A;
>
> 运行后输出的量:矩阵 A 的秩 RA 和各阶顺序主子式 $\det A_r$ $(r=1,2,\cdots,n)$ 的值 $h1$ 及其相关信息,A 能否进行 LU 分解的信息.

```
function hl =pdLUfj(A)
[n n] =size(A); RA =rank(A);
if RA ~ =n
  disp('请注意:因为 A 的 n 阶行列式 hl 等于零,所以 A 不能进行 LU 分解.A 的秩 RA 如下:')
  RA,hl =det(A);
  return
end
if RA = =n
  for p =1:n
      h(p) =det(A(1:p, 1:p));
  end
      hl =h(1:n);
```

```
        for i =1:n
          if h(1,i) = =0
            disp('请注意:因为 A 的 r 阶主子式等于零,所以 A 不能进行 LU 分解.A 的秩 RA 和各阶顺序主子式值 hl 依次如下:')
            hl;RA
          return
        end
      end
      if h(1,i) ~ =0
        disp('请注意:因为 A 的各阶主子式都不等于零,所以 A 能进行 LU 分解.A 的秩 RA 和各阶顺序主子式值 hl 依次如下:')
      hl;RA
        end
      end
```

例 3.4.1 判断下列矩阵能否进行 LU 分解,并求矩阵的秩.

$$(1)\begin{pmatrix}1&2&3\\1&12&7\\4&5&6\end{pmatrix};(2)\begin{pmatrix}1&2&3\\1&2&7\\4&5&6\end{pmatrix};(3)\begin{pmatrix}1&2&3\\1&2&3\\4&5&6\end{pmatrix}.$$

解 (1) 在 MATLAB 工作窗口输入程序

```
>>A =[1 2 3;1 12 7;4 5 6];hl =pdLUfj(A)
```

运行后输出结果为

请注意:因为 A 的各阶主子式都不等于零,所以 A 能进行 LU 分解.A 的秩 RA 和各阶顺序主子式值 hl 依次如下:

```
RA = 3, hl = 1   10   -48
```

(2) 在 MATLAB 工作窗口输入程序

```
>>A =[1 2 3;1 2 7;4 5 6];hl =pdLUfj(A)
```

运行后输出结果为

请注意:因为 A 的 r 阶主子式等于零,所以 A 不能进行 LU 分解.A 的秩 RA 和各阶顺序主子式值 hl 依次如下:

```
RA = 3, hl =1    0    12
```

(3) 在 MATLAB 工作窗口输入程序

```
>> A =[1 2 3;1 2 3;4 5 6];hl =pdLUfj(A)
```

运行后输出结果为

请注意:因为 A 的 n 阶行列式 hl 等于零,所以 A 不能进行 LU 分解.A 的秩 RA 如下:

```
RA = 2, hl = 0
```

3.4.2 直接 LU 分解法及其 MATLAB 程序

这里将无行交换的矩阵 A 的 LU 分解称为**矩阵的直接 LU 分解**.

（一）矩阵的直接 LU 分解的方法

设 n 阶矩阵 $\boldsymbol{A}$ 的各阶顺序主子式 $\det \boldsymbol{A}_r$ $(r=1,2,\cdots,n)$不为零，则 $\boldsymbol{A}$ 有唯一的 LU 分解(3.15)式. 由矩阵的乘法，得 $a_{1j}=u_{1j}$ $(j=1,2,\cdots,n)$. 当 $i>1$ 时，有

$$a_{ij}=\begin{cases}\sum_{k=1}^{j} l_{ik}u_{kj}, & \text{当 } j<i,\\ \sum_{k=1}^{i-1} l_{ik}u_{kj}+u_{ij}, & \text{当 } j\geqslant i.\end{cases} \tag{3.17}$$

例 3.4.2 用直接 LU 分解法分解矩阵 $\boldsymbol{A}=\begin{pmatrix}1&0&2&0\\0&1&0&1\\1&2&4&3\\0&1&0&3\end{pmatrix}$，并写出对应的 MATLAB 程序.

解 设

$$\begin{pmatrix}1&0&2&0\\0&1&0&1\\1&2&4&3\\0&1&0&3\end{pmatrix}=\begin{pmatrix}1&0&0&0\\l_{21}&1&0&0\\l_{31}&l_{32}&1&0\\l_{41}&l_{42}&l_{43}&1\end{pmatrix}\begin{pmatrix}u_{11}&u_{12}&u_{13}&u_{14}\\0&u_{22}&u_{23}&u_{24}\\0&0&u_{33}&u_{34}\\0&0&0&u_{44}\end{pmatrix}.$$

根据(3.17)式可以逐行逐列求出 l_{ij}和 u_{ij}，即

(1) 由 $a_{1j}=u_{1j}$ $(j=1,2,3,4)$，得 $u_{11}=1, u_{12}=0, u_{13}=2, u_{14}=0$. 对应的 MATLAB 程序

```
>> for j =1:4
        U(1,j) =A(1,j);
   end
```

(2) 由 $l_{i1}=a_{i1}/u_{11}$ $(i=2,3,4)$，得 $l_{21}=0, l_{31}=1, l_{41}=0$. 对应的 MATLAB 程序

```
>> for i =2:4
     L(i,1) =A(i,1)/U(1,1);
   end
```

(3) 由 $a_{2j}=l_{21}u_{1j}+u_{2j}$ $(j=2,3,4)$，得 $u_{2j}=a_{2j}-l_{21}u_{1j}$ $(j=2,3,4)$，$u_{22}=1$，$u_{23}=0, u_{24}=1$. 对应的 MATLAB 程序

```
>> for j =2:4
        U(2,j) =A(2,j) -L(2,1) * U(1,j);
   end
```

(4) 由 $a_{i2}=l_{i1}u_{12}+l_{i2}u_{22}$ $(i=3,4)$，得 $l_{i2}=(a_{i2}-l_{i1}u_{12})/u_{22}$ $(i=3,4)$，即 $l_{32}=2, l_{42}=1$. 对应的 MATLAB 程序

```
>> for i =3:4
        L(i,2) =(A(i,2) -L(i,1) * U(1,2))/U(2,2);
   end
```

(5) 由 $a_{3j}=l_{31}u_{1j}+l_{32}u_{2j}+u_{3j}$ $(j=3,4)$得

$$u_{3j}=a_{3j}-(l_{31}u_{1j}+l_{32}u_{2j})\ (j=3,4), \text{即 } u_{33}=2, u_{34}=1.$$

对应的 MATLAB 程序

```
>> for j =3:4
        U(3,j) =A(3,j) -L(3,1:2) * U(1:2,j);
   end
```

(6) 由 $a_{33}=l_{41}u_{13}+l_{42}u_{23}+l_{43}u_{33}$,得 $l_{43}=(a_{33}-l_{41}u_{13}+l_{42}u_{23})/u_{33}=0.$

对应的 MATLAB 程序

```
>> L(4,3) =(A(3,3) - L(4,1:2) * U(1:2,3))/U(3,3);
```

(7) 由 $a_{44}=l_{41}u_{14}+l_{42}u_{24}+l_{43}u_{34}+u_{44}$,得

$$u_{44}=a_{44}-(l_{41}u_{14}+l_{42}u_{24}+l_{43}u_{34})=2.$$

对应的 MATLAB 程序

```
>> U(4,4) =A(4,4) -L(4,1:3) * U(1:3,4);
```

故

$$\begin{pmatrix}1&0&2&0\\0&1&0&1\\1&2&4&3\\0&1&0&3\end{pmatrix}=\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&2&1&0\\0&1&0&1\end{pmatrix}\begin{pmatrix}1&0&2&0\\0&1&0&1\\0&0&2&1\\0&0&0&2\end{pmatrix}.$$

(二) 矩阵直接进行 LU 分解的 MATLAB 程序

根据定理 3.5 和(3.16)式编写 MATLAB 程序如下:

将矩阵 A 进行直接 LU 分解的 MATLAB 程序

输入的量:系数矩阵 A;

输出的量:矩阵 A 的秩 RA 和各阶顺序主子式 $\det A_r (r=1,2,\cdots,n)$ 的值 $h1$ 及其相关信息,A 能否进行 LU 分解的信息. 如果 A 能进行 LU 分解,则输出 L 和 U.

```
function hl =zhjLU(A)
  [n n] =size(A); RA =rank(A);
  if RA ~ =n
     disp('请注意:因为 A 的 n 阶行列式 hl 等于零,所以 A 不能进行 LU 分解.
A 的秩 RA 如下:'), RA,hl =det(A);
    return
  end
```

```
    if RA = =n
      for p =1:n
          h(p) =det(A(1:p, 1:p));
      end
        hl =h(1:n);
      for i =1:n
        if h(1,i) = =0
            disp('请注意:因为 A 的 r 阶主子式等于零,所以 A 不能进行 LU 分解.
A 的秩 RA 和各阶顺序主子式值 hl 依次如下:'), hl;RA
        return
      end
    end
    if h(1,i) ~ =0
      disp('请注意:因为 A 的各阶主子式都不等于零,所以 A 能进行 LU 分解.A 的
秩 RA 和各阶顺序主子式值 hl 依次如下:')
      for j =1:n
            U(1,j) =A(1,j);
      end
      for k =2:n
        for i =2:n
          for j =2:n
                L(1,1) =1;L(i,i) =1;
            if i >j
                L(1,1) =1;L(2,1) =A(2,1)/U(1,1); L(i,1) =A(i,1)/U(1,1);
                L(i,k) =(A(i,k) - L(i,1:k -1) * U(1:k -1,k))/U(k,k);
            else
                U(k,j) =A(k,j) -L(k,1:k -1) * U(1:k -1,j);
            end
          end
        end
      end
        hl;RA,U,L
    end
  end
```

例 3.4.3 用矩阵进行直接 LU 分解的 MATLAB 程序分解矩阵

$$A=\begin{pmatrix}1&0&2&0\\0&1&0&1\\1&2&4&3\\0&1&0&3\end{pmatrix}.$$

解 在 MATLAB 工作窗口输入程序

```
>> A=[1 0 2 0;0 1 0 1;1 2 4 3;0 1 0 3]; hl=zhjLU(A)
```

运行后输出结果

请注意:因为 A 的各阶主子式都不等于零,所以 A 能进行 LU 分解.A 的秩 RA 和各阶顺序主子式值 hl 依次如下:

```
RA =4                     L = 1  0  0  0
U = 1  0  2  0                0  1  0  0
    0  1  0  1                1  2  1  0
    0  0  2  1                0  1  0  1
    0  0  0  2            hl =1  1  2  4
```

由此可见,例 3.4.3 和例 3.4.2 中 A 的 LU 分解式相同.

3.4.3 含交换矩阵的 LU 分解法及其 MATLAB 程序

(一) 含交换矩阵的 LU 分解法

如果只知道 n 阶矩阵 A 可逆而不能保证 A 的各阶顺序主子式不为零,那么在消元过程中可能会遇到某个 $a_{kk}^{(k)}=0$ 的情况,但这时必至少有一个 $a_{ik}^{(k)}\neq 0$, $i=k+1,\cdots,n$,只需像列主元消元法那样,将第 k 行与第 i 行交换,就可以继续施行前面的消元过程,而这种行交换相当于 A 左乘一个由单位矩阵第 k 与第 i 行交换所得的初等交换矩阵 E_{ki},于是将 A 化为上三角形矩阵 U 的过程相当于 A 左乘一系列的初等交换矩阵和单位下三角形矩阵,可写作

$$M_{n-1}E_{k-1,i_{k-1}}\cdots M_1E_{1,i_1}A=U. \tag{3.18}$$

记 $E_{k-1,i_{k-1}}\cdots E_{1,i_1}=P$,$P$ 是单位矩阵经若干次行交换得到的**交换矩阵**,由(3.18)式可推出

$$MPA=U.$$

仍记 $M^{-1}=L$,就得到 $PA=LU$.

定理 3.6 若 n 阶矩阵 A 可逆,则存在交换矩阵 P 使

$$PA=LU \tag{3.19}$$

L,U 分别为单位下三角形矩阵和上三角形矩阵.

(二) 含交换矩阵的 LU 分解法的 MATLAB 程序

含交换矩阵的 LU 分解法的 MATLAB 程序是使用列主元消元法进行的,其命令和功能如下:

MATLAB 命令	功 能
[X U] = lu(A)	输出 X 为一交换矩阵 P^{-1} 与单位下三角形矩阵 L 之积，U 为上三角形矩阵，使得 $A=XU$，其中 $X=P^{-1}*L$.
[L U P] = lu(A)	输出 L 为单位下三角形矩阵，U 为上三角形矩阵，P 为一交换矩阵，使 $PA=LU$.

例 3.4.4 用两种方法将下列矩阵进行 LU 分解，其中

$$(1)\ A=\begin{pmatrix}0.1 & 2 & 3 & 4\\ -1 & 2 & -3 & 4\\ 11 & 21 & 13 & 41\\ 5 & 7 & 8 & 9\end{pmatrix};\qquad (2)\ B=\begin{pmatrix}0 & 2 & 4 & 1\\ 2 & 8 & 6 & 4\\ 3 & 10 & 8 & 8\\ 4 & 12 & 10 & 6\end{pmatrix}.$$

解 (1) 在 MATLAB 工作窗口输入程序

```
>> A=[0.1 2 3 4;-1 2 -3 4;11 21 13 41;5 7 8 9];[X U]=lu(A)
A=[0.1 2 3 4;-1 2 -3 4;11 21 13 41;5 7 8 9];[L U P]=lu(A)
```

运行后输出结果

```
X =   0.0091   0.4628 1.0000      0   U =11.0000 21.0000 13.0000 41.0000
     -0.0909   1.0000      0      0            0  3.9091 -1.8182  7.7273
      1.0000        0      0      0            0       0  3.7233  0.0512
      0.4545  -0.6512 0.2436 1.0000            0       0       0 -4.6171
L =   1.0000         0        0        0    P =0  0  1  0
     -0.0909    1.0000        0        0       0  1  0  0
      0.0091    0.4628   1.0000        0       1  0  0  0
      0.4545   -0.6512   0.2436   1.0000       0  0  0  1
```

使 $A=XU$ 和 $PA=LU$.

(2) 在 MATLAB 工作窗口输入程序

```
>> B=[0 2 4 1;2 8 6 4;3 10 8 8;4 12 10 6];[X U] = lu(B)
B=[0 2 4 1;2 8 6 4;3 10 8 8;4 12 10 6];  [L U P] = lu(B)
```

运行后输出结果

```
X =        0  1.0000  1.0000       0   U =4  12  10  6
      0.5000  1.0000       0       0      0   2   1  1
      0.7500  0.5000       0  1.0000      0   0   3  0
      1.0000       0       0       0      0   0   0  3
L =1.0000          0       0       0   P =0   0   0  1
      0.5000  1.0000       0       0      0   1   0  0
           0  1.0000  1.0000       0      1   0   0  0
      0.7500  0.5000       0  1.0000      0   0   1  0
```

使 $\boldsymbol{B}=\boldsymbol{XU}$ 和 $\boldsymbol{PB}=\boldsymbol{LU}$.

3.4.4 判断正定对称矩阵的方法及其 MATLAB 程序

定理 3.7 对称矩阵 $\boldsymbol{A}$ 为正定的充分必要条件是 $\boldsymbol{A}$ 的各阶顺序主子式都为正.

根据定理 3.7 编写的 MATLAB 程序如下:

> **判断矩阵 $\boldsymbol{A}$ 是否是正定对称矩阵的 MATLAB 程序**
>
> 输入的量:系数矩阵 $\boldsymbol{A}$;
>
> 输出的量:矩阵 $\boldsymbol{A}$ 的转置矩阵 $z\boldsymbol{A}$ 和 $\boldsymbol{A}$ 的各阶顺序主子式 $\det \boldsymbol{A}_r$ $(r=1,2,\cdots,n)$ 的值 $h1$ 及其相关信息,$\boldsymbol{A}$ 能否进行 LU 分解的信息.

```
function hl = zddc(A)
[n n] =size(A);
for p =1:n
  h(p) =det(A(1:p, 1:p));
end
  hl =h(1:n);zA =A';chaA =A - zA;
  if chaA = =0
    disp('请注意:A 是对称矩阵')
  else
    disp('请注意:A 不是对称矩阵')
  end
for i =1:n
  if h(1,i) < =0
      disp('请注意:因为 A 的各阶顺序主子式 hl 不全大于零,所以 A 不是正定
的.A 的转置矩阵 zA 和各阶顺序主子式值 hl 依次如下:'),
      hl;zA
      return
  end
end
if h(1,i) >0
    disp('请注意:因为 A 的各阶顺序主子式 hl 都大于零,所以 A 是正定的.A 的
转置矩阵 zA 和各阶顺序主子式值 hl 依次如下:')
    hl;zA
end
```

例 3.4.5 判断下列矩阵是否是正定对称矩阵:

$$(1)\begin{pmatrix} 0.1 & 2 & 3 & 4 \\ -1 & 2 & -3 & 4 \\ 11 & 21 & 13 & 41 \\ 5 & 7 & 8 & 9 \end{pmatrix};(2)\begin{pmatrix} 1 & -1 & 2 & 1 \\ -1 & 3 & 0 & -3 \\ 2 & 0 & 9 & -6 \\ 1 & -3 & -6 & 19 \end{pmatrix};$$

(3) $\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$；(4) $\begin{pmatrix} -2 & 1 & 1 \\ 1 & -6 & 0 \\ 1 & 0 & -4 \end{pmatrix}$.

解 (1) 在 MATLAB 工作窗口输入程序

```
>> A=[0.1 2 3 4;-1 2 -3 4;11 21 13 41;5 7 8 9];hl=zddc(A)
```

运行后输出结果

```
请注意:A不是对称矩阵
请注意:因为A的各阶顺序主子式hl不全大于零,所以A不是正定的.A的转置矩
阵zA和各阶顺序主子式值hl依次如下:
zA =1/10    -1        11       5
      2      2        21       7
      3     -3        13       8
      4      4        41       9
hl =1/10   11/5   -1601/10   3696/5
```

因此,A 既不是正定矩阵,也不是对称矩阵.

(2) 在 MATLAB 工作窗口输入程序

```
>> A=[1 -1 2 1;-1 3 0 -3;2 0 9 -6;1 -3 -6 19],hl=zddc(A)
```

运行后输出结果

```
A =  1   -1    2    1
    -1    3    0   -3
     2    0    9   -6
     1   -3   -6   19
请注意:A是对称矩阵
请注意:因为A的各阶顺序主子式hl都大于零,所以A是正定的.A的转置矩阵zA
和各阶顺序主子式值hl依次如下:
zA =  1   -1    2    1
     -1    3    0   -3
      2    0    9   -6
      1   -3   -6   19
hl =  1    2    6   24
```

(3) 在 MATLAB 工作窗口输入程序

```
>> A=[1/sqrt(2) -1/sqrt(2) 0 0; -1/sqrt(2) 1/sqrt(2) 0 0;0 0 1/
sqrt(2) -1/sqrt(2);0 0 -1/sqrt(2) 1/sqrt(2)], hl=zddc(A)
```

运行后输出结果

```
A = 985/1393    -985/1393     0            0
    -985/1393   985/1393      0            0
    0           0             985/1393     -985/1393
    0           0             -985/1393    985/1393
```

请注意:A 是对称矩阵

请注意:因为 A 的各阶顺序主子式 hl 不全大于零,所以 A 不是正定的.A 的转置矩阵 zA 和各阶顺序主子式值 hl 依次如下:

```
zA = 985/1393    -985/1393     0            0
     -985/1393   985/1393      0            0
     0           0             985/1393     -985/1393
     0           0             -985/1393    985/1393
hl = 985/1393    0             0            0
```

可见,$\boldsymbol{A}$ 不是正定矩阵,是半正定矩阵;因为 $\boldsymbol{A}=\boldsymbol{A}^{\mathrm{T}}$. 因此,$\boldsymbol{A}$ 是对称矩阵.

(4) 在 MATLAB 工作窗口输入程序

```
>> A=[-2 1 1;1 -6 0;1 0 -4];hl=zddc(A)
```

运行后输出结果

```
A = -2    1    1
     1   -6    0
     1    0   -4
```

请注意:A 是对称矩阵

请注意:因为 A 的各阶顺序主子式 hl 不全大于零,所以 A 不是正定的.A 的转置矩阵 zA 和各阶顺序主子式值 hl 依次如下:

```
zA = -2    1    1      hl = -2   11   -38
      1   -6    0
      1    0   -4
```

$\boldsymbol{A}$ 不是正定矩阵,是负定矩阵;因为 $\boldsymbol{A}=\boldsymbol{A}^{\mathrm{T}}$,因此,$\boldsymbol{A}$ 是对称矩阵.

3.4.5 正定对称矩阵的楚列斯基(Cholesky)分解及其 MATLAB 程序

一些实际问题如用有限元法作结构力学计算,常归结为求解系数矩阵为正定对称的线性方程组,这时形如(3.15)式的 LU 分解有更简化的形式.

定理 3.8 若 n 阶矩阵 $\boldsymbol{A}$ 是正定对称矩阵,则 $\boldsymbol{A}$ 可分解成对角元为正的上三角形矩阵 $\boldsymbol{R}$ 的转置矩阵与它自身之积,即

$$\boldsymbol{A}=\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} \tag{3.20}$$

或者表示为

$$\boldsymbol{A}=\boldsymbol{L}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{L} \tag{3.21}$$

其中 $\boldsymbol{R}$ 是单位上三角形矩阵,$\boldsymbol{D}$ 是对角元为正的对角矩阵. 这种分解称**三角分解或楚列斯基分解.**

n 阶正定或半正定的对称矩阵 A 的楚列斯基分解的 MATLAB 程序如下：

MATLAB 命令	功　能
R = chol(A)	输出 R 为上三角形矩阵，使 $A=R^TR$
[R,p] = chol(A)	输出 R 为上三角形矩阵，使 $R^TR=A(1:p-1,1:p-1)$

如果读者想了解关于 chol 更多的使用信息，可以在 MATLAB 工作窗口输入查询命令

```
>>help chol.
```

例 3.4.6 用两种方法将例 3.4.5 中的矩阵进行楚列斯基分解.

解 (1) 在 MATLAB 工作窗口输入程序

```
>> A=[0.1 2 3 4;-1 2 -3 4;11 21 13 41;5 7 8 9]; R1 =chol(A),[R,p]
=chol(A)
```

运行后输出结果

```
??? Error using = = > chol,Matrix must be positive definite.
```

即 A 不是正定矩阵，也不是半正定矩阵，又不是对称矩阵，不能进行楚列斯基分解.

(2) 在 MATLAB 工作窗口输入程序

```
>> A=[1 -1 2 1;-1 3 0 -3;2 0 9 -6;1 -3 -6 19]; R1 =chol(A),[R,
p] =chol(A)
```

运行后输出结果

```
R1 =1       -1          2           1
     0   1393/985   1393/985   -1393/985
     0       0      1351/780   -1351/390
     0       0          0           2
R =1       -1          2           1
    0   1393/985   1393/985   -1393/985
    0       0      1351/780   -1351/390
    0       0          0           2
p =0
```

(3) 在 MATLAB 工作窗口输入程序

```
>>  B=[1/sqrt(2) -1/sqrt(2) 0 0; -1/sqrt(2) 1/sqrt(2) 0 0; 0 0
1/sqrt(2) -1/sqrt(2); 0 0 -1/sqrt(2) 1/sqrt(2)]; R1 =chol(B), [R,p] =
chol(B),R'*R,B
```

运行后输出结果

```
R1 =0.8409  -0.8409       0        0   R =0.8409  -0.8409       0        0
         0   0.0000       0        0          0   0.0000       0        0
         0        0  0.8409  -0.8409          0        0  0.8409  -0.8409
         0        0       0   0.0000          0        0       0   0.0000
```

```
p = 0
B =   0.7071  -0.7071        0        0  B =   0.7071  -0.7071        0        0
     -0.7071   0.7071        0        0       -0.7071   0.7071        0        0
           0        0   0.7071  -0.7071             0        0   0.7071  -0.7071
           0        0  -0.7071   0.7071             0        0  -0.7071   0.7071
```

可见,虽然 $\boldsymbol{B}$ 不是正定矩阵,是半正定矩阵和对称矩阵,但也可以进行楚列斯基分解,且 $\boldsymbol{B}=\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}$.

(4) 在 MATLAB 工作窗口输入程序

```
>> A =[ -2 1 1;1 -6 0;1 0 -4];R1 =chol(A),[R,p] =chol(A)
```

运行后输出结果

```
??? Error using = = > chol,Matrix must be positive definite.
```

由输出的结果可以看出,虽然 $\boldsymbol{A}$ 是对称矩阵,但 $\boldsymbol{A}$ 既不是正定矩阵,也不是半正定矩阵,所以 $\boldsymbol{A}$ 不能进行楚列斯基分解.

综上所述,**正定矩阵或半正定矩阵的对称矩阵可以进行楚列斯基分解,反之不然.**



习题 3.4

1. 判断下列矩阵能否进行 LU 分解,如果可以进行 LU 分解,则求 $\boldsymbol{L},\boldsymbol{U}$ 使 $\boldsymbol{A}=\boldsymbol{LU}$,并求矩阵的秩.

(1) $\begin{pmatrix} 2 & 4 & -6 \\ 1 & 5 & 3 \\ 1 & 3 & 2 \end{pmatrix}$;(2) $\begin{pmatrix} 1 & 1 & 6 \\ -1 & 2 & 9 \\ 1 & -2 & 3 \end{pmatrix}$.

2. 判断下列矩阵能否进行 LU 分解,如果可以进行 LU 分解,则用两种方法将下列矩阵进行 LU 分解,其中

(1) $\boldsymbol{A}=\begin{pmatrix} 0.2 & 2 & 3 & 4 \\ -1 & 0.1 & -3 & 4 \\ 11 & 21 & 13 & 41 \\ 5 & 7 & 8 & 9 \end{pmatrix}$;(2) $\boldsymbol{B}=\begin{pmatrix} 0 & 26 & 4 & 1 \\ 12 & 8 & 6 & 4 \\ 3 & 10 & 8 & 8 \\ 4 & 12 & 10 & 6 \end{pmatrix}$

3. 判断下列矩阵是否是正定对称矩阵,如果是正定对称矩阵,将矩阵进行楚列斯基分解.

(1) $\begin{pmatrix} 1 & -1 & 2 & 1 \\ -1 & 3 & 0 & -3 \\ 2 & 0 & 9 & -6 \\ 1 & -3 & -6 & 19 \end{pmatrix}$; (2) $\begin{pmatrix} 2 & 0 & -2 \\ 0 & 4 & 0 \\ -2 & 0 & 5 \end{pmatrix}$; (3) $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}$;

(4) $\begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 1 \\ -1 & 0 & 1 & 1 \end{pmatrix}$.

3.5 求解线性方程组的 LU 方法及其 MATLAB 程序

将矩阵 $\boldsymbol{A}$ 的 LU 分解后，代入矩阵方程(3.9)中，得

$$\boldsymbol{LUX}=\boldsymbol{b}. \tag{3.22}$$

求(3.9)式的解等价与求下面两个方程组的解

$$\begin{cases} \boldsymbol{Ly}=\boldsymbol{b}, \\ \boldsymbol{UX}=\boldsymbol{y}. \end{cases} \tag{3.23}$$

即

$$\begin{cases} y_1 & & & & = b_1, \\ l_{21}y_1 + & y_2 & & & = b_2, \\ l_{31}y_1 + & l_{32}y_2 + & y_3 & & = b_3, \\ \cdots\cdots\cdots\cdots \\ l_{n1}y_1 + & l_{n2}y_2 + & l_{n3}y_3 + \cdots + l_{n,n-1}y_{n-1} & + y_n & = b_n, \end{cases}$$

$$\begin{cases} u_{11}x_1 + & u_{12}x_2 + & u_{13}x_3 + & u_{14}x_4 + & \cdots & + u_{1n}x_n & = y_1, \\ & u_{22}x_2 + & u_{23}x_3 + & u_{24}x_4 + & \cdots & + u_{2n}x_n & = y_2, \\ & & u_{33}x_3 + & u_{34}x_4 + & \cdots & + u_{3n}x_n & = y_3, \\ & & & u_{44}x_4 + & \cdots & + u_{4n}x_n & = y_4, \\ & & & & \ddots & \vdots & \vdots \\ & & & & & u_{nn}x_n & = y_n. \end{cases}$$

求得递推公式为

$$\begin{cases} y_i = b_i - \sum_{k=1}^{i-1} l_{ik}y_k \quad i = 1,2,\cdots,n, \\ x_i = \dfrac{1}{u_{ii}}\left(y_i - \sum_{k=i+1}^{n} l_{ik}x_k\right) \quad i = n,n-1,\cdots,2,1, \text{且 } k \leqslant n. \end{cases} \tag{3.24}$$

3.5.1 求解线性方程组的楚列斯基分解法及其 MATLAB 程序

若矩阵方程 $\boldsymbol{AX}=\boldsymbol{b}$ 的 n 阶系数矩阵 $\boldsymbol{A}$ 是正定对称矩阵，则 $\boldsymbol{A}$ 可分解成对角元为正的上三角形矩阵 $\boldsymbol{R}$ 的转置矩阵与它自身之积，即

$$\boldsymbol{A}=\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}. \tag{3.25}$$

将(3.25)式代入(3.9)中,得 $R^{T}RX=b$,故

$$X=R^{-1}(R^{-1})^{T}b \tag{3.26}$$

这种解矩阵方程 $AX=b$ 的方法称为**楚列斯基分解法**.

例3.5.1 先将矩阵 A 进行楚列斯基分解,然后解矩阵方程 $AX=b$,并用其他方法验证.

$$A=\begin{pmatrix}1 & -1 & 2 & 1\\ -1 & 3 & 0 & -3\\ 2 & 0 & 9 & -6\\ 1 & -3 & -6 & 19\end{pmatrix}, b=\begin{pmatrix}1\\3\\5\\7\end{pmatrix}.$$

解 根据(3.26)式编写MATLAB程序,然后在工作窗口输入

```
>>A=[1 -1 2 1;-1 3 0 -3;2 0 9 -6;1 -3 -6 19];
b1=1:2:7; b=b1'; R=chol(A);C=A-R'*R,R1=inv(R);R2=R1';
x=R1*R2*b,Rx=A\b-x
```

运行后输出方程组的解和验证结果

```
x =            Rx = 1.0e-014 *   C = 1.0e-015 *
     -8.0000        -0.7105          0              0  0  0
      0.3333        -0.0833          0        -0.4441  0  0
      3.6667         0.2220          0              0  0  0
      2.0000         0.1332          0              0  0  0
```

3.5.2 求解线性方程组的直接LU分解法及其MATLAB程序

若矩阵方程 $AX=b$ 的 n 阶系数矩阵 A 是可以直接分解为一个单位下三角形矩阵 L 和一个上三角形矩阵 U 的积 $A=LU$,将 LU 代入 $AX=b$ 中,得 $LUX=b$,故

$$X=U^{-1}L^{-1}b \tag{3.27}$$

这种解矩阵方程 $AX=b$ 的方法称为**直接LU分解法**.

例3.5.2 首先将矩阵 A 直接进行LU分解,然后解矩阵方程 $AX=b$

$$A=\begin{pmatrix}1 & 0 & 2 & 0\\ 0 & 1 & 0 & 1\\ 1 & 2 & 4 & 3\\ 0 & 1 & 0 & 3\end{pmatrix}, b=\begin{pmatrix}1\\2\\-1\\5\end{pmatrix}.$$

解 (1) 首先将矩阵 A 直接进行LU分解.在MATLAB工作窗口输入程序

```
>> A=[1 0 2 0;0 1 0 1;1 2 4 3;0 1 0 3];b=[1;2;-1;5]; h1=zhjLU(A),
```

运行后输出LU分解

请注意:因为A的各阶主子式都不等于零,所以A能进行LU分解.A的秩RA和各

阶顺序主子式值 hl 依次如下:

```
RA  =4                    L =1  0  0  0
 U  =1  0  2  0              0  1  0  0
        0  1  0  1           1  2  1  0
        0  0  2  1           0  1  0  1
        0  0  0  2        hl =1  1  2  4
```

$\boldsymbol{A}$ 分解为一个单位下三角形矩阵 $\boldsymbol{L}$ 和一个上三角形矩阵 $\boldsymbol{U}$ 的积 $\boldsymbol{A}=\boldsymbol{LU}$.

(2) 然后根据(3.27)式编写 MATLAB 程序,然后在工作窗口输入

```
>>  U =[1 0 2 0;0 1 0 1;0 0 2 1;0 0 0 2];
    L =[1 0 0 0;0 1 0 0;1 2 1 0;0 1 0 1];
    b =[1;2; -1;5];U1 =inv(U); L1 =inv(L); X =U1 * L1 * b
```

运行后输出方程组的解

```
X =   8.50000000000000
      0.50000000000000
     -3.75000000000000
      1.50000000000000
```

3.5.3 求解线性方程组的选主元的 LU 方法及其 MATLAB 程序

(一) 若矩阵方程 $\boldsymbol{AX}=\boldsymbol{b}$ 的 n 阶系数矩阵 $\boldsymbol{A}$ 可以分解为

$$\boldsymbol{PA}=\boldsymbol{LU},$$

其中 $\boldsymbol{L}$ 是一个单位下三角形矩阵,$\boldsymbol{U}$ 是一个上三角形矩阵,$\boldsymbol{P}$ 是由单位矩阵经若干次行交换得到的交换矩阵. 将 $\boldsymbol{LU}$ 代入 $\boldsymbol{AX}=\boldsymbol{b}$ 中,则原方程的同解方程为 $\boldsymbol{PAX}=\boldsymbol{Pb}$,即 $\boldsymbol{LUX}=\boldsymbol{Pb}$,解得

$$\boldsymbol{X}=\boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{Pb} \tag{3.28}$$

(二) 若矩阵方程 $\boldsymbol{AX}=\boldsymbol{b}$ 的 n 阶系数矩阵 $\boldsymbol{A}$ 可以分解为

$$\boldsymbol{A}=\boldsymbol{FU},$$

其中 $\boldsymbol{F}$ 是一个单位下三角形矩阵与交换矩阵的积,$\boldsymbol{U}$ 是一个上三角形矩阵,将 $\boldsymbol{FU}$ 代入 $\boldsymbol{AX}=\boldsymbol{b}$ 中,原方程的同解方程为 $\boldsymbol{FUX}=\boldsymbol{b}$,解得

$$\boldsymbol{X}=\boldsymbol{U}^{-1}\boldsymbol{F}^{-1}\boldsymbol{b}. \tag{3.29}$$

用(3.28)式或(3.29)式解矩阵方程 $\boldsymbol{AX}=\boldsymbol{b}$ 的方法称为**选主元的 LU 方法.**

例 3.5.3 先将矩阵 $\boldsymbol{A}$ 进行 LU 分解,然后解矩阵方程 $\boldsymbol{AX}=\boldsymbol{b}$,其中

$$\boldsymbol{A}=\begin{pmatrix}0.1 & 2 & 3 & 4\\ -1 & 2 & -3 & 4\\ 11 & 21 & 13 & 41\\ 5 & 7 & 8 & 9\end{pmatrix},\boldsymbol{b}=\begin{pmatrix}1\\ 2\\ -1\\ 5\end{pmatrix}.$$

解 方法 1 根据(3.28)式编写 MATLAB 程序,然后在工作窗口输入

```
>> A=[0.1 2 3 4;-1 2 -3 4;11 21 13 41;5 7 8 9];
b=[1;2;-1;5];[L U P]=lu(A),U1=inv(U);
L1=inv(L);X=U1*L1*P*b
```

运行后输出结果

```
L =  1.0000        0        0        0      P=0  0  1  0
    -0.0909   1.0000        0        0        0  1  0  0
     0.0091   0.4628   1.0000        0        1  0  0  0
     0.4545  -0.6512   0.2436   1.0000        0  0  0  1
U =11.0000  21.0000  13.0000  41.0000     X=-1.2013
          0   3.9091  -1.8182   7.7273        3.3677
          0        0   3.7233   0.0512        0.0536
          0        0        0  -4.6171       -1.4440
```

方法 2 根据(3.29)式编写 MATLAB 程序,然后在工作窗口输入

```
>> A=[0.1 2 3 4;-1 2 -3 4;11 21 13 41;5 7 8 9];
b=[1;2;-1;5];[F U]=lu(A),U1=inv(U);F1=inv(F);X=U1*F1*b
```

运行后输出结果

```
F=  0.0091   0.4628   1.0000        0   U=11.0000  21.0000  13.0000  41.0000
   -0.0909   1.0000        0        0           0   3.9091  -1.8182   7.7273
    1.0000        0        0        0           0        0   3.7233   0.0512
    0.4545  -0.6512   0.2436   1.0000           0        0        0  -4.6171
X =[-1.2013 3.3677 0.0536 -1.4440]'
```

例 3.5.4 投入产出综合平衡分析.

国民经济各个部门之间存在着相互依存的关系,每个部门在运转中将其他部门的产品或半成品经过加工(称为**投入**)变为自己的产品(称为**产出**),如何根据各部门间的投入－产出关系,确定各部门的产出水平,以满足社会的需求,是投入产出综合平衡模型研究的课题.试讨论如下的简化问题.

设国民经济仅由农业、制造业和服务业三个部门构成,已知某年它们之间的投入产出关系、外部需求、初始投入等如表 3－3 所示(数字表示产值,单位为亿元).

表 3－3 国民经济各个部门间的关系

产出／投入	农业	制造业	服务业	外部需求	总产出
农业	25	20	30	35	110
制造业	30	10	45	115	200
服务业	20	60	/	70	150
初始投入	35	110	75		
总投入	110	200	150		

表中第一行数字表示,农业总产出为 110 亿元,其中 25 亿元农产品用于农业生产本身(如提供种子),20 亿元用于制造业(如提供木材、毛皮),30 亿元用于服务业,剩下 35 亿元农产品用来满足外部需求(包括消费、积累、出口等). 可以类似地解释第二、三行数字. 第一列数字中,25 亿元如前所述,30 亿元是制造业对农业的投入(如提供农具),20 亿元是服务业对农业的投入,35 亿元的初始投入包括工资、税收、进口等,总投入 110 亿元与总产出相等.

假定每个部门的产出与投入是成正比的,由表 3-3 和 MATLAB 程序

```
>> nn=25/110,zn=30/110,fn=20/110,cn=35/110,
nzt=nn+zn+fn+cn,
nz=20/200,zz=10/200,fz=60/200,cz=110/200,
zzt=nz+zz+fz+cz,
nf=30/150,zf=45/150,ff=0/150,cf=75/150,fzt=nf+zf+ff+cf
```

能够计算出这三个部门的投入产出表,如表 3-4.

表 3-4 投入产出表

产出 / 投入	农 业	制 造 业	服 务 业
农业	0.227 3	0.100 0	0.200 0
制造业	0.272 7	0.050 0	0.300 0
服务业	0.181 8	0.300 0	0

表 3-4 中第一行、第二列的数字 0.100 0 表示,生产 1 个单位产值的制造业产品需投入 0.100 0 个单位产值的农产品,这是由表 3-3 中 20 亿元农产品投入制造业,可以产出 200 亿元制造业总产值而来的(20 ÷ 200 = 0.1). 同样,第三行、第一列的数字 0.181 8 表示,生产 1 个单位产值的农产品需要 0.181 8 个单位的服务业产值,因为表 3-3 中 20 亿元的服务业产值投入农业,得到 110 亿元的农业总产值. 表 3-4 的数字称为投入系数或消耗系数,如果技术水平没有变化,可以假设投入系数是常数.

(1) 设有 n 个部门,已知投入系数,给定外部需求,建立求解各部门总产出的模型.

(2) 设投入系数如表 3-4 所给,如果今年对农业、制造业和服务业的外部需求分别为 50,150,110 亿元,问这三个部门的总产出分别应为多少.

(3) 如果三个部门的外部需求分别增加 1 个单位,它们的总产出应分别增加多少.

(4) 如果对于任意给定的、非负的外部需求,都能得到非负的总产出,模型就称为可行的. 问为使模型可行,投入系数应满足什么条件.

解 (1) 若共有 n 个部门,记一定时期内第 i 个部门的总产出为 x_i,其中对第 j 个部门的投入为 x_{ij},满足的外部需求为 d_i,则

$$x_i = \sum_{j=1}^{n} x_{ij} + d_i \quad (i = 1,2,\cdots,n). \tag{3.30}$$

表 3-3 的每一行即满足(3.30)式. 记第 j 个部门的单位产出需要第 i 个部门的投入为 a_{ij},在每个部门的产出与投入成正比的假定下,有

$$a_{ij} = x_{ij}/x_j \quad (i,j = 1,2,\cdots,n). \tag{3.31}$$

表 3-4 中的投入系数即为 a_{ij},(3.31)式代入(3.30)式得

$$x_i = \sum_{j=1}^{n} a_{ij}x_j + d_i \quad (i = 1,2,\cdots,n). \tag{3.32}$$

记投入系数矩阵 $\boldsymbol{A} = (a_{ij})_{n\times n}$,产出向量 $\boldsymbol{X} = (x_1,\cdots,x_n)^{\mathrm{T}}$,需求向量 $\boldsymbol{D} = (d_1,\cdots,d_n)^{\mathrm{T}}$,则(3.32)式写作

$$\boldsymbol{X} = \boldsymbol{AX} + \boldsymbol{D}, \tag{3.33}$$

或

$$(\boldsymbol{E} - \boldsymbol{A})\boldsymbol{X} = \boldsymbol{D}. \tag{3.34}$$

若$(\boldsymbol{E} - \boldsymbol{A})$可逆,则

$$\boldsymbol{X} = (\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{D}. \tag{3.35}$$

当投入系数 $\boldsymbol{A}$ 和外部需求 $\boldsymbol{D}$ 给定后,即可算出各部门的总产出 $\boldsymbol{X}$.

(2) 表 3-4 中的投入系数及对三个部门的外部需求,可输入 MATLAB 程序

```
>> A =[0.2273 0.1 0.2;0.2727 0.05 0.3; 0.1818 0.3 0];
D =[50 150 110]';
B = eye(3) -A; C = inv(B), X = B \D
```

运行后输出结果

```
C =                                X =
    1.4805   0.2754   0.3787          156.9994
    0.5633   1.2676   0.4929          272.5308
    0.4382   0.4304   1.2167          220.3017
```

得到三个部门的总产出分别应为 156.999 4,272.530 8,220.301 7(亿元).

(3)(3.35)式表明总产出 $\boldsymbol{X}$ 对外部需求 $\boldsymbol{D}$ 是线性的,所以当 $\boldsymbol{D}$ 增加 1 个单位时,$\boldsymbol{X}$ 的增量由$(\boldsymbol{E} - \boldsymbol{A})^{-1}$决定. 在上面程序中已经算出 $\boldsymbol{C} = (\boldsymbol{E} - \boldsymbol{A})^{-1}$,其第一列数字表明,当对农业的需求增加 1 个单位时,农业、制造业和服务业的总产出应分别增加 1.480 5,0.563 3,0.438 2 单位. 其余类似. 这些数字称部门关联系数.

(4) 要使模型可行,即对任意的外部需求 $\boldsymbol{D} \geqslant 0$,由(3.35)式能够得到总产出 $\boldsymbol{X} \geqslant 0$,显然只$(\boldsymbol{E} - \boldsymbol{A})^{-1} \geqslant 0$(指每个元素非负,下同).

因为 $A\geqslant 0,(E-A)(E+A+A^2+\cdots+A^k)=E-A^{k+1}$,

所以只要 $A^k\to 0$ $(k\to\infty)$,就有 $(E-A)^{-1}=\sum_{k=0}^{\infty}A^k\geqslant 0$. 由矩阵范数定义可知 $A^k\to 0$ 与 $\|A^k\|\to 0$ 等价,且 $\|A^k\|\leqslant\|A\|^k$,故只要 $\|A\|_1<1$ (这里取便于应用的 1 范数),即

$$\sum_{i=1}^{n}a_{ij}<1\ (j=1,2,\cdots,n) \tag{3.36}$$

模型就是可行的.

如果投入系数 A 是根据实际数据算出的(如由表 3-3 得到表 3-4),由(3.31)式可知(3.36)式等价于

$$\sum_{i=1}^{n}x_{ij}<x_j\quad(j=1,2,\cdots,n) \tag{3.37}$$

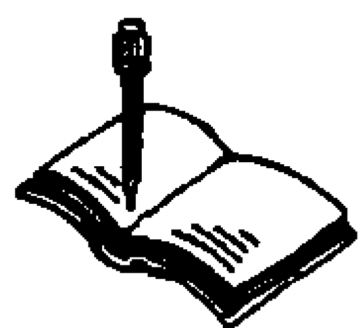
可以看出,只要初始投入非负,(3.37)式是自然成立的.

(三) 解线性方程组 $A_{m\times n}X=b$ 的选主元的 LU 方法,编写 MATLAB 程序如下:

> **用 LU 分解法解线性方程组 $A_{m\times n}X=b$ 的 MATLAB 程序**
>
> 输入的量:系数矩阵 A 和常数向量 b;
>
> 输出的量:系数矩阵 A 和增广矩阵 B 的秩 RA,RB, 方程组中未知量的个数 n 和方程组解及其有关的信息.

```
function [RA,RB,n,X,Y] = LUjfcz(A,b)
    [n n] = size(A);B = [A b]; RA = rank(A); RB = rank(B);
for p = 1:n
    h(p) = det(A(1:p, 1:p));
end
    h1 = h(1:n);
for i = 1:n
    if h(1,i) = = 0
        disp('请注意:因为 A 的 r 阶主子式等于零,所以 A 不能进行 LU 分解.A
的秩 RA 和各阶顺序主子式值 h1 依次如下:')
      h1;RA
      return
  end
end
if h(1,i) ~ = 0
        disp('请注意:因为 A 的各阶主子式都不等于零,所以 A 能进行 LU 分解.
A 的秩 RA 和各阶顺序上子式值 h1 依次如下:')
```

```
X = zeros(n,1); Y = zeros(n,1); C = zeros(1,n);r = 1:1;
for p = 1:n - 1
    [max1,j] = max(abs(A(p:n,p))); C = A(p,:);
    A(p,:) = A(j + P1,:); C = A(j + P1,:);
    g = r(p); r(p) = r(j + P1); r(j + P1) = g;
    for k = p + 1:n
        H = A(k,p)/A(p,p); A(k,p) = H;
        A(k,p + 1:n) = A(k,p + 1:n) - H* A(p,p + 1:n);
    end
end
Y(1) = B(r(1));
for k = 2:n
  Y(k) = B(r(k)) - A(k,1:k - 1) * Y(1:k - 1);
end
  X(n) = Y(n)/A(n,n);
for i = n - 1: - 1:1
              X(i) = (Y(i) - A(i, i + 1:n) * X (i + 1:n))/A(i,i);
end
end
[RA,RB,n,X,Y]';
```



习 题 3.5

1. 先将矩阵 B 进行楚列斯基分解，然后解矩阵方程 $Bx = b$：

(1) $B = \begin{pmatrix} 1 & -1 & 2 & 1 \\ -1 & 3 & 0 & -3 \\ 2 & 0 & 9 & -6 \\ 1 & -3 & -6 & 19 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \end{pmatrix}$; (2) $B = \begin{pmatrix} 12 & -3 & 2 & 1 \\ -3 & 23 & -7 & -3 \\ 2 & -7 & 99 & -6 \\ 1 & -3 & -6 & 19 \end{pmatrix}, b = \begin{pmatrix} 6 \\ 3 \\ -16 \\ 7 \end{pmatrix}$.

2. 先将矩阵 A 进行 LU 分解，然后解矩阵方程 $AX = b$：

(1) $A = \begin{pmatrix} 0.02 & 7 & 3 & 4 \\ -1 & 0.12 & -3 & 4 \\ 11 & 21 & 13 & 41 \\ 5 & 7 & 8 & 9 \end{pmatrix}, b = \begin{pmatrix} 0.5 \\ 1.2 \\ -1 \\ 5 \end{pmatrix}$;

(2) $A = \begin{pmatrix} \frac{1}{3} & -2 & 76 & \frac{3}{4} & 5 \\ 3 & \frac{1}{\sqrt{3}} & 0 & -7 & 89 \\ 56 & 0 & -1 & 3 & 13 \\ 21 & 65 & -7 & 8 & 15 \\ 23 & 76 & 51 & 62 & 81 \end{pmatrix}, b = \begin{pmatrix} \frac{2}{\sqrt{5}} \\ -2 \\ 3 \\ 51 \\ \frac{5}{\sqrt{71}} \end{pmatrix}$.

3. 解线性方程组:

(1) $\begin{cases} 12x_1 - 3x_2 + 3x_3 + 4x_4 = 15, \\ -18x_1 + 3x_2 - x_3 - x_4 = -15, \\ x_1 + x_2 + x_3 + x_4 = 6, \\ 3x_1 + x_2 - x_3 + x_4 = 2; \end{cases}$ (2) $\begin{cases} 2x_1 + x_2 = 3 \\ x_1 + 2x_2 - 3x_3 = -3, \\ 3x_2 - 7x_3 + 4x_4 = -10, \\ 2x_3 + 5x_4 = 2. \end{cases}$

3.6 误差分析及其两种 MATLAB 程序

对于实际问题导出的线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的系数矩阵 $\boldsymbol{A}$ 和右端向量 $\boldsymbol{b}$ 往往带有误差(扰动),下面讨论当 $\boldsymbol{A}$ 或 $\boldsymbol{b}$ 有微小变化对解 $\boldsymbol{X}$ 的影响.

3.6.1 误差分析

(一) 两个例子

例 3.6.1 解下列矩阵方程 $\boldsymbol{AX}=\boldsymbol{b}$,并比较方程(1)和(2)有何区别,它们的解有何变化. 其中

(1) $\boldsymbol{A}=\begin{pmatrix}2 & 2\\ 2 & 2.001\end{pmatrix}, \boldsymbol{b}=\begin{pmatrix}4\\4\end{pmatrix}$; (2) $\boldsymbol{A}=\begin{pmatrix}2 & 2\\ 2 & 2.001\end{pmatrix}, \boldsymbol{b}=\begin{pmatrix}4\\4.01\end{pmatrix}$.

解 (1) 在 MATLAB 工作窗口输入程序

```
>>A=[2 2;2 2.001];b=[4;4];X=A\b
```

运行后输出方程的解为 $\boldsymbol{X}=(2.000\quad 0.000)^{\mathrm{T}}$.

(2) 在 MATLAB 工作窗口输入程序

```
>>A=[2 2;2 2.001];b=[4;4.01];X=A\b
```

运行后输出方程的解为 $\boldsymbol{X}=(-8.000\ 10.000)^{\mathrm{T}}$.

方程(1)和(2)的系数矩阵相同,常数向量 $\boldsymbol{b}=\begin{pmatrix}4\\4\end{pmatrix}$ 和 $\boldsymbol{b}=\begin{pmatrix}4\\4.01\end{pmatrix}$ 的差为 $\delta\boldsymbol{b}=\begin{pmatrix}0\\-0.01\end{pmatrix}$,则 $\boldsymbol{Ax}=\boldsymbol{b}$ 的解 $\begin{pmatrix}2\\0\end{pmatrix}$ 和 $\begin{pmatrix}-8\\10\end{pmatrix}$ 的差为 $\delta\boldsymbol{X}=\begin{pmatrix}10\\-10\end{pmatrix}$. $\boldsymbol{b}$ 的微小变化,引起 $\boldsymbol{X}$ 的很大变化,即 $\boldsymbol{X}$ 对 $\boldsymbol{b}$ 的**扰动是敏感的**. 从方程组本身看,是由于 $\boldsymbol{A}$ 接近奇异,求解过程中 $\boldsymbol{A}$ 用很小的数做除数的缘故.

例 3.6.2 解下列矩阵方程 $\boldsymbol{AX}=\boldsymbol{b}$,并比较方程(1)和(2)有何区别,它们的解有何变化. 其中

(1) $A=\begin{pmatrix}1&1/2&1/3&1/4&1/5&1/6&1/7\\1/2&1/3&1/4&1/5&1/6&1/7&1/8\\1/3&1/4&1/5&1/6&1/7&1/8&1/9\\1/4&1/5&1/6&1/7&1/8&1/9&1/10\\1/5&1/6&1/7&1/8&1/9&1/10&1/11\\1/6&1/7&1/8&1/9&1/10&1/11&1/12\\1/7&1/8&1/9&1/10&1/11&1/12&1/13\end{pmatrix},b=\begin{pmatrix}1\\2\\2\\2\\2\\2\\2\end{pmatrix}$;

(2) $A=\begin{pmatrix}1.001&1/2&1/3&1/4&1/5&1/6&1/7\\1/2&1/3&1/4&1/5&1/6&1/7&1/8\\1/3&1/4&1/5&1/6&1/7&1/8&1/9\\1/4&1/5&1/6&1/7&1/8&1/9&1/10\\1/5&1/6&1/7&1/8&1/9&1/10&1/11\\1/6&1/7&1/8&1/9&1/10&1/11&1/12\\1/7&1/8&1/9&1/10&1/11&1/12&1/13\end{pmatrix},b=\begin{pmatrix}1\\2\\2\\2\\2\\2\\2\end{pmatrix}$.

解 (1) 矩阵方程 $AX=b$ 的系数矩阵 A 为 7 阶希尔伯特(Hilbert)矩阵，我们可以用下列命令计算 n 阶希尔伯特矩阵

```
>>h=hilb(n)        % 输出h为n阶希尔伯特矩阵
```

在 MATLAB 工作窗口输入程序

```
>> A=hilb(7);b=[1;2;2;2;2;2;2];X=A\b
```

运行后输出 $AX=b$ 的解为 $X=(-35\quad 504\quad -1260\quad -4200\quad 20790\quad -27720\quad 12012)^T$.

(2) 在 MATLAB 工作窗口输入程序

```
>>B =[0.001,zeros(1,6);zeros(6,1),zeros(6,6)];
A=(B+hilb(7)); b=[1;2;2;2;2;2;2];X=A\b
```

运行后输出方程的解为 $X=(-33\quad 465\quad -966\quad -5181\quad 22409\quad -29015\quad 12413)^T$.

在 MATLAB 工作窗口输入程序

```
>> X =[-33,465,-966,-5181,22409,-29015,12413]';
X1 =[-35,504,-1260,-4200,20790,-27720,12012]'; wu=X1'-X'
```

运行后输出方程(1)和(2)的解的误差为

$$\delta X=(-2\quad 39\quad -294\quad 981\quad -1619\quad 1295\quad -401)^T.$$

方程(1)和(2)的系数矩阵的差为 $\delta A=\begin{pmatrix}0.001&O_{1\times6}\\O_{6\times1}&O_{6\times6}\end{pmatrix}$，常数向量相同，则 $Ax=b$ 的解的差为 $\delta X=(-2\quad 39\quad -294\quad 981\quad -1619\quad 1295\quad -401)^T$. A 的微小变化，引起 X 的很大变化，即 X 对 A 的扰动是敏感的.

（二）误差分析

定义 3.4 如果系数矩阵 $\boldsymbol{A}$ 或常数向量 $\boldsymbol{b}$ 的微小变化，引起方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的解 $\boldsymbol{X}$ 的很大变化，则称此**方程组是病态的**，也称**系数矩阵 $\boldsymbol{A}$ 是病态的**，否则称**此方程组是良态的，系数矩阵 $\boldsymbol{A}$ 是良态的**.

可以想到，几乎接近奇异的矩阵大概是病态的. 为了定量地估计 $\boldsymbol{X}$ 对 $\boldsymbol{b}$ 或 $\boldsymbol{A}$ 的扰动敏感的程度，需要度量向量和矩阵"大小"的数量指标. 向量范数和矩阵范数正是这样的指标，它们分别用 $\|\boldsymbol{X}\|$，$\|\boldsymbol{b}\|$，$\|\boldsymbol{A}\|$ 表示. **矩阵 $\boldsymbol{A}$ 的条件数**

$$\mathrm{Cond}(\boldsymbol{A})=\|\boldsymbol{A}^{-1}\|\cdot\|\boldsymbol{A}\|$$

是方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的解 $\boldsymbol{X}$ 对系数矩阵 $\boldsymbol{A}$ 或常数向量 $\boldsymbol{b}$ 有微小变化时敏感性的一种度量，为了证明这个问题，首先证明下面的定理.

定理 3.9 如果矩阵 $\boldsymbol{B}$ 的范数 $\|\boldsymbol{B}\|<1$，$\boldsymbol{E}$ 为单位矩阵，则 $\boldsymbol{E}\pm\boldsymbol{B}$ 为非奇异矩阵，且

$$\|(\boldsymbol{E}\pm\boldsymbol{B})^{-1}\|\leqslant\frac{1}{1-\|\boldsymbol{B}\|}.$$

证明（用反证法） 如果 $\boldsymbol{E}\pm\boldsymbol{B}$ 为奇异矩阵，即 $\det(\boldsymbol{E}\pm\boldsymbol{B})=0$，则 $(\boldsymbol{E}\pm\boldsymbol{B})\boldsymbol{X}=\boldsymbol{O}$ 有非零解，即存在 $\boldsymbol{X}_0\neq\boldsymbol{O}$ 使得 $\boldsymbol{BX}_0=\pm\boldsymbol{X}_0$ 成立，从而 $\|\boldsymbol{B}\|\geqslant1$，与假设 $\|\boldsymbol{B}\|<1$ 矛盾. 又由 $(\boldsymbol{E}\pm\boldsymbol{B})(\boldsymbol{E}\pm\boldsymbol{B})^{-1}=\boldsymbol{E}$，有

$$(\boldsymbol{E}\pm\boldsymbol{B})^{-1}=\boldsymbol{E}\mp\boldsymbol{B}(\boldsymbol{E}\pm\boldsymbol{B})^{-1},$$

即 $\|(\boldsymbol{E}\pm\boldsymbol{B})^{-1}\|\leqslant\|\boldsymbol{E}\|+\|\boldsymbol{B}\|\cdot\|(\boldsymbol{E}\pm\boldsymbol{B})^{-1}\|$，

$$\|(\boldsymbol{E}\pm\boldsymbol{B})^{-1}\|\leqslant\frac{1}{1-\|\boldsymbol{B}\|}.$$

定理 3.10 设线性方程组

$$\boldsymbol{AX}=\boldsymbol{b}\ (\boldsymbol{b}\neq\boldsymbol{0}) \tag{3.38}$$

的系数矩阵 $\boldsymbol{A}_{n\times n}$ 是非奇异矩阵，

（1）如果条件数 $\mathrm{Cond}\ (\boldsymbol{A})_p$ 很大时，即 $\mathrm{Cond}(\boldsymbol{A})_p\gg1$，则(3.38)为病态的；

（2）如果条件数 $\mathrm{Cond}\ (\boldsymbol{A})_p$ 相对较小时，则(3.38)为良态的；

（3）如果 $\boldsymbol{A}$ 不变，$\boldsymbol{b}$ 的绝对误差为 $\delta\boldsymbol{b}$，解 $\boldsymbol{X}$ 的绝对误差为 $\delta\boldsymbol{X}$，则 $\boldsymbol{X}$ 的相对误差为

$$\frac{\|\delta\boldsymbol{X}\|}{\|\boldsymbol{X}\|}\leqslant\mathrm{Cond}\ (\boldsymbol{A})\cdot\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}; \tag{3.39}$$

（4）如果 $\boldsymbol{b}$ 不变，$\boldsymbol{A}$ 的绝对误差为 $\delta\boldsymbol{A}$，解 $\boldsymbol{X}$ 的绝对误差为 $\delta\boldsymbol{X}$，则 $\boldsymbol{X}$ 的相对误差为

$$\frac{\|\delta\boldsymbol{X}\|}{\|\boldsymbol{X}+\delta\boldsymbol{X}\|}\leqslant\mathrm{Cond}\ (\boldsymbol{A})\cdot\frac{\|\delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|}. \tag{3.40}$$

如果 $\|\boldsymbol{A}^{-1}\|\cdot\|\delta\boldsymbol{A}\|<1$，则 $\boldsymbol{X}$ 的相对误差为

$$\frac{\|\delta X\|}{\|X\|}\leqslant\frac{\mathrm{Cond}(A)\dfrac{\|\delta A\|}{\|A\|}}{1-\mathrm{Cond}(A)\dfrac{\|\delta A\|}{\|A\|}}. \tag{3.41}$$

证明 (1) 设 A 不变,若 b 的误差(扰动)δb 引起 X 的误差(扰动)为 δX. 于是有 $A(X+\delta X)=b+\delta b$,即 $A\delta X=\delta b$,可得

$$\delta X=A^{-1}\delta b, \tag{3.42}$$

从而

$$\|\delta X\|\leqslant\|A^{-1}\|\cdot\|\delta b\|. \tag{3.43}$$

另一方面,由(3.38)有 $\|b\|\leqslant\|A\|\cdot\|X\|$,即

$$\|X\|\geqslant\frac{\|b\|}{\|A\|}. \tag{3.44}$$

联合(3.43)式和(3.44)式得到

$$\frac{\|\delta X\|}{\|X\|}\leqslant\mathrm{Cond}\ (A)\cdot\frac{\|\delta b\|}{\|b\|}.$$

可见,X 对 b 扰动的敏感程度取决于 A 的条件数 Cond (A). A 的条件数可作为误差的上界,即 X 的(相对)误差可以达到 b 的(相对)误差的 Cond(A)倍. 显然,条件数越大,(由 b 的误差引起的)X 的误差可能越大.

(2) 设 b 不变,可类似地考虑 A 的误差(扰动)δA 引起 X 的误差(扰动)δX,即$(A+\delta A)(X+\delta X)=b$,从而 $\delta X=-A^{-1}\cdot\delta A\cdot(X+\delta X)$得 $\|\delta X\|\leqslant\|A^{-1}\|\cdot\|\delta A\|\cdot\|X+\delta X\|$,

即
$$\frac{\|\delta X\|}{\|X+\delta X\|}\leqslant\|A^{-1}\|\cdot\|A\|\cdot\frac{\|\delta A\|}{\|A\|},$$

就是
$$\frac{\|\delta X\|}{\|X+\delta X\|}\leqslant\mathrm{Cond}\ (A)\cdot\frac{\|\delta A\|}{\|A\|},$$

所以
$$(A+\delta A)\delta X=-(\delta A)X. \tag{3.45}$$

如果 δA 不受限制的话,$A+\delta A$ 可能是奇异的,而

$$A+\delta A=A(E+A^{-1}\delta A).$$

由定理 3.9 知,当 $\|A^{-1}\delta A\|<1$ 时,$(E+A^{-1}\delta A)^{-1}$存在,由(3.45)式得

$$\delta X=-(E+A^{-1}\delta A)^{-1}A^{-1}(\delta A)X,$$

所以
$$\|\delta X\|\leqslant\frac{\|A^{-1}\|\ \|\delta A\|\ \|X\|}{1-\|A^{-1}(\delta A)\|}.$$

设 $\|A^{-1}\|\cdot\|\delta A\|<1$,即得

$$\frac{\|\delta X\|}{\|X\|}\leqslant\frac{\|A^{-1}\|\cdot\|A\|\cdot\dfrac{\|\delta A\|}{\|A\|}}{1-\|A^{-1}\|\cdot\|A\|\cdot\dfrac{\|\delta A\|}{\|A\|}},$$

即为(3.41)式. 证毕

如果 $\|\delta A\|$ 充分小,且在 $\|A^{-1}\|\cdot\|\delta A\|<1$ 的条件下,则(3.41)式说明矩阵 A 的相对误差 $\frac{\|\delta A\|}{\|A\|}$ 在解中可能放大 Cond(A)倍.

总之,Cond(A)越小,由系数矩阵 A 或常数向量 b 的相对误差引起方程组(3.38)的解 X 的相对误差就越小,此时方程组(3.38)是良态的,系数矩阵 A 是良态的. 反之,如果条件数 $\text{Cond}(A)_p$ 越大,即 $\text{Cond}(A)_p \gg 1$,由系数矩阵 A 或常数向量 b 的相对误差引起方程组(3.38)的解 X 的相对误差就可能越大,此时方程组(3.38)是病态的,系数矩阵 A 是病态的,也越难得到方程组(3.38)的比较准确的解.

3.6.2 求 P 条件数和讨论 $AX=b$ 解的性态的 MATLAB 程序

根据定理 3.10 可知,判断线性方程组(3.38)是否病态需要计算条件数,在第一章中我们已经给出了计算矩阵 A 的 2 范数条件数、1 范数条件数、∞ 范数条件数、弗罗贝尼乌斯(Frobenius)范数条件数、条件数倒数等 MATLAB 命令. 用这些命令计算各种条件数非常方便. 现编写了下面求 P 条件数和讨论 $AX=b$ 解的性态的 MATLAB 程序.

求 P 条件数和讨论 $AX=b$ 解的性态的 MATLAB 程序

输入的量:系数矩阵 A;

输出的量:系数矩阵 A 的∞条件数,1 条件数,2 条件数,弗罗贝尼乌斯条件数和方程组解的性态的有关的信息.

```
function Acp = zpjxpb(A)
    Acw = cond (A, inf);Ac1 = cond (A,1);
    Ac2 = cond (A,2);Acf = cond (A,'fro');dA = det(A);
if (Acw >50)&(dA <0.1)
    disp('请注意:AX = b 是病态的,A 的∞ 条件数,1 条件数,2 条件数,Frobe-
nius 条件数和 A 的行列式的值依次如下:')
    Acp = [Acw Ac1 Ac2 Acf dA]';
else
    disp(' AX = b 是良态的,A 的∞ 条件数,1 条件数,2 条件数,Frobenius 条
件数和 A 的行列式的值依次如下:')
    Acp = [Acw Ac1 Ac2 Acf dA]';
end
```

n 阶希尔伯特矩阵定义为 $H=\begin{pmatrix} 1 & 1/2 & \cdots & 1/n \\ 1/2 & 1/3 & \cdots & 1/(n+1) \\ \vdots & \vdots & & \vdots \\ 1/n & 1/(n+1) & \cdots & 1/(2n-1) \end{pmatrix}$,当 n 很

大时呈严重病态,常作为研究病态现象的系数矩阵.

例 3.6.3 根据定理 3.10,讨论线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 解的性态,并且求出 $\boldsymbol{A}$ 的 4 种条件数. 其中

(1) $\boldsymbol{A}$ 为 7 阶希尔伯特矩阵;(2) $\boldsymbol{A}=\begin{pmatrix}2 & 3 & -1 & 5\\ 3 & 1 & 2 & -7\\ 4 & 1 & -3 & 6\\ 1 & -2 & 4 & -7\end{pmatrix}$.

解 (1)首先将求 P 条件数和讨论 $\boldsymbol{AX}=\boldsymbol{b}$ 解的性态的 MATLAB 程序保存名为 zpjxpb.m 的 M 文件,然后在 MATLAB 工作窗口输入程序

```
>> Acp =zpjxpb(hilb(7)); Acp',det(hilb(7))
```

运行后输出结果

```
请注意:AX =b 是病态的,A 的∞ 条件数,1 条件数,2 条件数,Frobenius 条件数和
A 的行列式的值依次如下:
    ans =   1.0e+008 *
        9.8519    9.8519    4.7537    4.8175    0.0000
    ans =   4.8358e-025
```

(2) 在 MATLAB 工作窗口输入程序

```
>> A=[2 3 -1 5;3 1 2 -7;4 1 -3 6;1 -2 4 -7];Acp=zpjxpb(A); Acp'
```

运行后输出结果

```
AX =b 是良态的,A 的∞ 条件数,1 条件数,2 条件数,Frobenius 条件数和 A 的行
列式的值依次如下:
    ans =
        14.1713   19.4954    8.2085   11.4203   327.0000
```

3.6.3 用 P 范数讨论 $\boldsymbol{AX}=\boldsymbol{b}$ 的解和 $\boldsymbol{A}$ 的性态的 MATLAB 程序

可以用下面编写的 MATLAB 程序讨论线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ (其中 $\boldsymbol{A}$ 是 n 阶希尔伯特矩阵)解的性态,并利用(3.39)式和(3.40)式讨论当 $\boldsymbol{A}$ 或 $\boldsymbol{b}$ 有变化时,$\boldsymbol{AX}=\boldsymbol{b}$ 的解、$\boldsymbol{A}$ 或 $\boldsymbol{b}$ 及其相对误差.

用 P 范数讨论 $\boldsymbol{AX}=\boldsymbol{b}$ 解和 $\boldsymbol{A}$ 的性态的 MATLAB 程序

输入的量:系数矩阵 $\boldsymbol{A}$、$\boldsymbol{A}$ 的近似矩阵 $j\boldsymbol{A}=\boldsymbol{A}+\delta\boldsymbol{A}$,常数向量 $\boldsymbol{b}$,$\boldsymbol{b}$ 的近似向量 $j\boldsymbol{b}=\boldsymbol{b}+\delta\boldsymbol{b}$ 和 P,其中 $P = 1,2,\mathrm{inf}$, 或 'fro.'.

输出的量:系数矩阵 $\boldsymbol{A}$ 的相对误差 xA,$\boldsymbol{AX}=\boldsymbol{b}$ 的解 X 和 $(\boldsymbol{A}+\delta\boldsymbol{A})\boldsymbol{X}=\boldsymbol{b}$ 或 $\boldsymbol{AX}=\boldsymbol{b}+\delta\boldsymbol{b}$ 的解 $\boldsymbol{wX}=\boldsymbol{X}+\delta\boldsymbol{X}$,$\boldsymbol{X}+\delta\boldsymbol{X}$ 和 $\boldsymbol{A}$ 的绝对误差 jxX 和 jxA, 解的绝对误差限 $jxxX$, $\boldsymbol{A}$ 的行列式的秩和 P 条件数及其方程组解的性态的有关的信息.

```
function Acp = zpjwc(A,jA,b,jb,P)
    Acp = cond (A,P);dA = det(A); X = A \b;
    dertaA = A - jA;
    PndA = norm(dertaA, P);dertab = b - jb;
    Pndb = norm(dertab, P);
if Pndb > 0
    jX = A \jb; Pnb = norm(b, P);PnjX = norm(jX,P); dertaX - X - jX;
    PnjdX = norm(dertaX, P);jxX = PnjdX /PnjX; PnjX = norm(jX,P);
    PnX = norm(X,P); jxX = PnjdX /PnjX; xX = PnjdX /PnX;
    Pndb = norm(dertab,P);
    xAb = Pndb /Pnb;Pnbj = norm(jb,P); xAbj = Pndb /Pnbj;
    Xgxx = Acp * xAb;
end
if PndA > 0
    jX = jA \b; dertaX = X - jX;PnX = norm(X,P);
    PnjdX = norm(dertaX, P);
    PnjX = norm(jX,P); jxX = PnjdX /PnjX;xX = PnjdX /PnX;
    PnjA = norm(jA,P); PnA = norm(A,P);
    PndA = norm(dertaA,P);xAbj = PndA /PnjA;xAb = PndA /PnA;
    Xgxx = Acp * xAb;
end
if (Acp >50)&(dA <0.1)
    disp('请注意:AX = b 是病态的,A 的 P 条件数 Acp,A 的行列式值 dA,解 X,
近似解 jX,解的相对误差 jxX,解的相对误差估计值 Xgxx,b 或 A 的相对误差 xAb 依次如
下:')
    Acp,dA,X',jX',xX',jxX',Xgxx',xAb',xAbj'
else
    disp('请注意: AX = b 是良态的,A 的 P 条件数 Acp,A 的行列式值 dA,解 X,
近似解 jX,解的相对误差 jxX,解的相对误差估计值 Xgxx,b 或 A 的相对误差 xAb 依次如
下:')
Acp,dA,X',jX',xX',jxX',Xgxx',xAb',xAbj'
end
```

例 3.6.4 根据定理 3.10,讨论线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 解的性态,并利用(3.40)式讨论当 $\boldsymbol{A}$ 的每个元都取 4 位有效数字时,其解的相对误差. 其中 $\boldsymbol{A}$ 为 7 阶希尔伯特矩阵,$\boldsymbol{b}=\begin{pmatrix}1 & \frac{1}{3} & 4 & 2 & 2 & 2 & 2\end{pmatrix}^{\mathrm{T}}$.

解 (1)取 ∞ 范数和 ∞ 条件数,线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的 $\boldsymbol{b}$ 不变时,取 ∞ 范数和 ∞ 条件数,系数矩阵 $\boldsymbol{A}$ 为 7 阶希尔伯特矩阵,$\boldsymbol{A}$ 中的每个元取 4 位有效数字,即

$$
jA=\begin{pmatrix}
1.0000 & 0.5000 & 0.3333 & 0.2500 & 0.2000 & 0.1667 & 0.1429\\
0.5000 & 0.3333 & 0.2500 & 0.2000 & 0.1667 & 0.1429 & 0.1250\\
0.3333 & 0.2500 & 0.2000 & 0.1667 & 0.1429 & 0.1250 & 0.1111\\
0.2500 & 0.2000 & 0.1667 & 0.1429 & 0.1250 & 0.1111 & 0.1000\\
0.2000 & 0.1667 & 0.1429 & 0.1250 & 0.1111 & 0.1000 & 0.0909\\
0.1667 & 0.1429 & 0.1250 & 0.1111 & 0.1000 & 0.0909 & 0.0833\\
0.1429 & 0.1250 & 0.1111 & 0.1000 & 0.0909 & 0.0833 & 0.0769
\end{pmatrix}.
$$

用 P 范数讨论 $AX=b$ 解和 A 的性态的 MATLAB 程序保存名为 zpjwc.m 的文件，然后在工作窗口输入 MATLAB 程序

```
>> jA =[1.0000  0.5000  0.3333  0.2500  0.2000  0.1667  0.1429
        0.5000  0.3333  0.2500  0.2000  0.1667  0.1429  0.1250
        0.3333  0.2500  0.2000  0.1667  0.1429  0.1250  0.1111
        0.2500  0.2000  0.1667  0.1429  0.1250  0.1111  0.1000
        0.2000  0.1667  0.1429  0.1250  0.1111  0.1000  0.0909
        0.1667  0.1429  0.1250  0.1111  0.1000  0.0909  0.0833
        0.1429  0.1250  0.1111  0.1000  0.0909  0.0833  0.0769];
   A =hilb(7);b =[1;1/3;4;2;2;2;2];
   jb =[1;0.3333;4;2;2;2;2]; Acp =zpjwc(A,jA,b,jb,inf)
```

运行后输出结果

请注意：AX = b 是病态的，A 的 P 条件数 Acp，A 的行列式值 dA，解 X，近似解 jX，解的相对误差 jxX，解的相对误差估计值 Xgxx，b 或 A 的相对误差 xAb 依次如下：

```
   Acp =                    dA =
      9.8519e+008              4.8358e-025
   ans =
     1.0e+007 *
   0.0020   -0.0697   0.6243   -2.3054   4.0677   -3.4123   1.0943
   ans =
     1.0e+004 *
     0.0349   -0.4807   2.1126   -5.1087   7.6557   -6.3239   2.1112
   xX =                 jxX =                  Xgxx =
      0.9981               530.3248                4.9291e+004
   xAb =                 xAbj =
      5.0032e-005           5.0031e-005
```

由此可见，因为∞ 条件数 $\mathrm{Cond}(A)_{\infty}\approx 985\ 194\ 889.719\ 848\ 3>>1$，所以此方程组为病态的 $AX=b$ 的解为

$X=(19\ 565,-697\ 256,6\ 243\ 300,-2.305\ 380,40\ 676\ 790,-34\ 123\ 320,10\ 942\ 932)^{\mathrm{T}}$,

$(jA)X=b$ 的解为

$$jX=(349,-48\ 079,21\ 126,-51\ 087,76\ 557,-63\ 239,21\ 112)^{T}.$$

解的相对误差

$$\frac{\|\delta X\|_{\infty}}{\|X\|_{\infty}}\approx 0.998\ 12,\frac{\|\delta X\|_{\infty}}{\|X+\delta X\|_{\infty}}\approx 530.32,\frac{\|\delta A\|_{\infty}}{\|A\|_{\infty}}\approx 5.003\ 2\times 10^{-5},$$

$$\frac{\|\delta X\|_{\infty}}{\|X+\delta X\|_{\infty}}\leqslant \mathrm{Cond}(A)_{\infty}\cdot\frac{\|\delta A\|_{\infty}}{\|A\|_{\infty}}\approx 49\ 291.27,$$

即相对误差放大了约 985 194 889.72 倍.

(2) 如果取 2 范数和 2 条件数计算,在 MATLAB 工作窗口输入程序

```
>> Acp =zpjwc(A,jA,b,jb,2)
```

运行后输出结果

请注意:AX = b 是病态的,A 的 P 条件数 Acp,A 的行列式值 dA,解 X,近似解 jX,解的相对误差 jxX,解的相对误差估计值 Xgxx,b 或 A 的相对误差 xAb 依次如下:

```
Acp =                    dA =
  4.7537e+008              4.8358e-025
ans =
  1.0e+007 *
  0.0020   -0.0697   0.6243   -2.3054   4.0677   -3.4123   1.0943
ans =
  1.0e+004 *
  0.0349   -0.4807   2.1126   -5.1087   7.6557   -6.3239   2.1112
xX =                     jxX =                    Xgxx =
   0.9981                   511.0640                 2.9951e+004
xAb =                    xAbj =
  6.3006e-005               6.3005e-005
```

因为 2 条件数 $\mathrm{Cond}(A)_2\approx 475\ 367\ 356.59>>1$,所以此方程组为病态的.解的相对误差

$$\frac{\|\delta X\|_2}{\|X\|_2}\approx 0.998\ 1,\frac{\|\delta A\|_2}{\|A\|_2}\approx 6.300\ 5\times 10^{-5},\frac{\|\delta X\|_2}{\|X+\delta X\|_2}\approx 511.06,$$

$$\frac{\|\delta X\|_2}{\|X+\delta X\|_2}\leqslant \mathrm{Cond}(A)_2\cdot\frac{\|\delta A\|_2}{\|A\|_2}\approx 29\ 950.85.$$

即相对误差放大了约 475 367 356.59 倍.

例 3.6.5 用 1 条件数,根据定理 3.10 讨论例 3.6.1(1)线性方程组 $AX=b$ 解的性态,并利用(3.39)式讨论当 b 的绝对误差为 $\delta b=(0\quad 0.01)^{T}$ 时,其解的相对误差.

解 (1) 在 MATLAB 工作窗口输入程序

```
>> A=[2 2;2 2.001]; jA=A; b=[4;4];
```

```
jb =[4;4.01]; Acp =zpjwc(A,jA,b,jb,1)
```

运行后输出结果

```
请注意:AX = b 是病态的,A 的 P 条件数 Acp,A 的行列式值 dA,解 X,近似解 jX,解的相对误差 jxX,解的相对误差估计值 Xgxx,b 或 A 的相对误差 xAb 依次如下:
Acp =                    dA =            ans =
  8.0040e+003              0.0020           2.0000    0.0000
ans =                    xX =            jxX =
  -8.0000   10.0000        10.0000          1.1111
Xgxx =                   xAb =           xAbj =
  10.0050                  0.0012           0.0012
```

因为 1 条件数 $\mathrm{Cond}(\boldsymbol{A})_1 \approx 8\ 004 >> 1$,所以此方程组为病态的. 当此方程组解和近似解等如上所示. 当常数向量 $\boldsymbol{b}$ 有微小误差 $\delta\boldsymbol{b} = (0 \quad 0.01)^{\mathrm{T}}$ 时,则 $\boldsymbol{b}$ 和解 $\boldsymbol{X}$ 的相对误差限分别为$\frac{\|\delta\boldsymbol{b}\|_1}{\|\boldsymbol{b}\|_1} \approx 0.001\ 2$,由(3.39)式估计

$$\frac{\|\delta\boldsymbol{X}\|_1}{\|\boldsymbol{X}\|_1} \leqslant \mathrm{Cond}(\boldsymbol{A})_1 \cdot \frac{\|\delta\boldsymbol{b}\|_1}{\|\boldsymbol{b}\|_1} \approx 10.005\ 0,$$

即相对误差放大了约 8 004 倍. $\frac{\|\delta\boldsymbol{X}\|_1}{\|\boldsymbol{X}\|_1} = 1.111\ 1$ 与估计的误差限 10.005 0 有差距.



习 题 3.6

1. 设 $\boldsymbol{A} = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$,当 $\boldsymbol{b} = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$有微小误差 $\delta\boldsymbol{b} = \begin{pmatrix} 0.1 \\ -0.1 \\ 0.1 \\ -0.1 \end{pmatrix}$时,估计方程组 $\boldsymbol{Ax} = \boldsymbol{b}$ 解的变化.

2. 解下列矩阵方程 $\boldsymbol{AX} = \boldsymbol{b}$,并比较方程(1)和(2)有何区别,它们的解有何变化. 其中

(1) $\boldsymbol{A} = \begin{pmatrix} 1 & 1 \\ 1 & 1.001 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$;(2) $\boldsymbol{A} = \begin{pmatrix} 1 & 1 \\ 1 & 1.001 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 3 \\ 3.01 \end{pmatrix}$.

3. 根据定理 3.10 讨论第 2 题的线性方程组 $\boldsymbol{AX} = \boldsymbol{b}$ 解的性态,并利用(3.39)式讨论当 $\boldsymbol{b}$ 的绝对误差为 $\delta\boldsymbol{b} = (0 \quad 0.01)^{\mathrm{T}}$ 时,其解的相对误差.

4. 根据定理 3.10,用∞范数讨论线性方程组 $\boldsymbol{AX} = \boldsymbol{b}$(其中 $\boldsymbol{A}$ 为 4 阶希尔伯特矩阵)解的性态,并利用(3.40)式讨论 $\boldsymbol{A}$ 中每个元都取 4 位有效数字时 (见下面的矩阵),其解的相对误差. 其中

$$A \approx \begin{pmatrix} 1.0000 & 0.5000 & 0.3333 & 0.2500 \\ 0.5000 & 0.3333 & 0.2500 & 0.2000 \\ 0.3333 & 0.2500 & 0.2000 & 0.1667 \\ 0.2500 & 0.2000 & 0.1667 & 0.1429 \end{pmatrix}, \quad b = \begin{pmatrix} 1.93 \\ 1.12 \\ 0.81 \\ 2.21 \end{pmatrix}.$$

5. 通过试验观察、分析下列问题是否病态，结果的计算误差与理论分析是否相符.

(1) $A = \begin{pmatrix} 3.021 & 2.714 & 6.913 \\ 1.031 & -4.273 & 1.121 \\ 5.084 & -5.832 & 9.155 \end{pmatrix}, b = \begin{pmatrix} 12.648 \\ -2.121 \\ 8.407 \end{pmatrix}$，求解矩阵方程 $AX = b$；将 $A(2,2)$ 改为 -4.275，再解 $AX = b$；

(2) $A = \begin{pmatrix} 3.01 & 6.03 & 1.99 \\ 1.27 & 4.16 & -1.23 \\ 0.987 & -4.81 & 9.34 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$，求解矩阵方程 $AX = b$；分别将 $A(1,1)$ 改为 3.00，$A(3,1)$ 改为 0.99，再解 $AX = b$；

(3) A 为 n 阶希尔伯特矩阵，令 x 为全 1 向量，算出 b；令 b 变化 1%，求解 x，分析误差 $(n = 30 \sim 50)$.

第四章 解线性方程组的迭代法

解线性方程组 $AX=b$ 的迭代法是从初始解出发，根据设计好的步骤用逐次求出的近似解逼近精确解. 在第三章中介绍的解线性方程组的直接方法一般适合于 A 为低阶稠密矩阵（指 n 不大且元多为非零）的情况，而在工程技术和科学计算中常会遇到**大型稀疏矩阵**（指 n 很大且零元较多）的方程组，迭代法在计算和存贮两方面都适合后一种情况. 由于迭代法是通过逐次迭代来逼近方程组的解，所以收敛性和收敛速度是构造迭代法时应该注意的问题. 另外，因为不同的系数矩阵具有不同的性态，所以大多数迭代方法都具有一定的适用范围. 有时，某种方法对于一类方程组迭代收敛，而对另一类方程组迭代时就发散. 因此，我们应该学会针对具有不同性质的线性方程组构造不同的迭代. 本章主要介绍迭代法的一些基本理论和雅可比（Jacobi）迭代法，高斯－塞德尔（Gauss-Seidel）迭代法，超松弛迭代法及其 MATLAB 程序，并对迭代的收敛性作简单说明.

4.1 迭代法和敛散性及其 MATLAB 程序

4.1.1 迭代法的思想

先用一个例子说明迭代法的思想.

例 4.1.1 求解方程组

$$\begin{cases} 10x_1 - x_2 - 2x_3 = 7.2, \\ -x_1 + 10x_2 - 2x_3 = 8.3, \\ -x_1 - x_2 + 5x_3 = 4.2. \end{cases} \tag{4.1}$$

解 设 $A=\begin{pmatrix} 10 & -1 & -2 \\ -1 & 10 & -2 \\ -1 & -1 & 5 \end{pmatrix}$, $X=\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$, $b=\begin{pmatrix} 7.2 \\ 8.3 \\ 4.2 \end{pmatrix}$，则方程组(4.1)为 $AX=b$. 将方程组(4.1)改写为

$$\begin{cases} x_1 = 0.1x_2 + 0.2x_3 + 0.72, \\ x_2 = 0.1x_1 + 0.2x_3 + 0.83, \\ x_3 = 0.2x_1 + 0.2x_2 + 0.84, \end{cases}$$

或写为 $X=BX+f$, (4.2)

其中

$$B=\begin{pmatrix}0 & 0.1 & 0.2\\ 0.1 & 0 & 0.2\\ 0.2 & 0.2 & 0\end{pmatrix},\quad f=\begin{pmatrix}0.72\\ 0.83\\ 0.84\end{pmatrix},\quad D=\begin{pmatrix}10 & & \\ & 10 & \\ & & 5\end{pmatrix},$$

显然,$A=D(E-B)$,则 $B=E-D^{-1}A$,$f=D^{-1}b$. (4.2)形式的好处是,将一组 x_i 代入右端,可立即从左端得到另一组 $x_i(i=1,2,3)$,如果这两组 x_i 相等,那么它就是方程组的解. 而当它们不等时,即可形成迭代过程. 任意取初始值 $X^{(0)}=(0,0,0)^{\mathrm{T}}$,代入右端后从左端得到 $X^{(1)}=(0.72,0.83,0.84)^{\mathrm{T}}$,再将这组值代入右端计算,如此下去,得到方程组(4.2)的迭代形式

$$\begin{cases}x_1^{(k+1)}=0.1x_2^{(k)}+0.2x_3^{(k)}+0.72,\\ x_2^{(k+1)}=0.1x_1^{(k)}+0.2x_3^{(k)}+0.83,\\ x_3^{(k+1)}=0.2x_1^{(k)}+0.2x_2^{(k)}+0.84\end{cases}\quad(k=0,1,2,\cdots).\tag{4.3}$$

将(4.3)简记为

$$X^{(k+1)}=BX^{(k)}+f.$$

其中 k $(k=0,1,2,\cdots)$表示迭代次数. 当迭代至 $k=9$ 次,可得

$$X^{(9)}=(1.0999,1.1999,1.2999)^{\mathrm{T}},$$

在 MATLAB 工作窗口输入程序

```
>>A=[10 -1 -2;-1 10 -2;-1 -1 5];b=[7.2;8.3;4.2];x=A\b
```

运行后输出方程组(4.1)的精确解 $X^*=(1.1000,1.2000,1.3000)^{\mathrm{T}}$,可见,$X^{(9)}$ 与 X^* 非常接近. 如果记 $R^{(9)}=X^{(9)}-X^*$,在 MATLAB 工作窗口输入程序

```
>> R9=[1.1-1.099;1.2-1.199;1.3-1.2999];nR9=norm(R9,inf)
```

运行后输出结果

```
nR9 = 0.0010
```

即,$\|R^{(9)}\|_\infty=0.0010$. 从此例可见,由迭代法产生的迭代向量序列$\{X^{(k)}\}$逐步逼近方程组 $AX=b$ 的精确解 X^*.

对于由线性方程组 $AX=b$ 通过恒等变形得到的任何一个等价方程组的迭代形式 $X^{(k+1)}=BX^{(k)}+f$,做出的迭代向量序列$\{X^{(k)}\}$不是一定逐步逼近 $AX=b$ 的精确解,请看下面的例题.

例 4.1.2 讨论由方程组(4.1)的第 2 个,第 3 个、第 1 个方程分离出 x_1,x_2,x_3 得到的等价方程组的迭代形式

$$\begin{cases}x_1^{(k+1)}=10x_2^{(k)}-2x_3^{(k)}-8.3,\\ x_2^{(k+1)}=-x_1^{(k)}+5x_3^{(k)}-4.2,\\ x_3^{(k+1)}=5x_1^{(k)}-0.5x_2^{(k)}-3.6\end{cases}\quad(k=0,1,2,\cdots),\tag{4.4}$$

做出的迭代向量序列$\{X^{(k)}\}$是否逐步逼近(4.1)的精确解 X^*.

解 取初始值 $X^{(0)}=(0,0,0)^T$，代入(4.4)式的右端，从左端得到 $X^{(1)}=(-8.3,-4.2,-3.6)^T$. 再将这组值代入(4.4)式的右端计算，得 $X^{(2)}=(-43.1,-13.9,-43)^T$. 继续迭代下去已无必要，因为由程序

```
>> R1 =[1.1 +8.3;1.2 +4.2;1.3 +3.6];nR1 =norm(R1,inf)
   R2 =[1.1 +43.1;1.2 +13.9;1.3 +43];nR2 =norm(R2,inf)
```

运行后得到误差向量的范数 $\|R^{(1)}\|_\infty=\|X^{(1)}-X^*\|_\infty=9.4000$，$\|R^{(2)}\|_\infty=\|X^{(2)}-X^*\|_\infty=44.3000$ 越来越大，可见迭代形式(4.4)是发散的，即由(4.4)式做出的迭代向量序列 $\{X^{(k)}\}$ 不逼近(4.1)式的精确解 X^*.

由例 4.1.1 和 例 4.1.2 可见，由同一个方程组(4.1)得到的两个等价方程组的迭代形式(4.3)和(4.4)，前者做出的迭代向量序列 $\{X^{(k)}\}$ 逐步逼近(4.1)式的精确解 X^*，而后者则不然. 因此，所有的迭代法都需要讨论收敛性和误差估计问题.

设线性方程组

$$AX=b \tag{4.5}$$

有唯一解 X^*，则

$$X^*=BX^*+f. \tag{4.6}$$

由(4.5)得到一个等价方程组的迭代公式

$$X^{(k+1)}=BX^{(k)}+f\quad(k=1,2,\cdots). \tag{4.7}$$

任意取一个初始向量 $X^{(0)}$ 代入(4.7)式，求出 $X^{(1)}$，将 $X^{(1)}$ 再代入(4.7)式，求出 $X^{(2)}$，如此反复进行求(4.5)式的近似解的方法称为**迭代法**. 如果 $\lim\limits_{k\to\infty}X^{(k)}=X^*$ 存在，称此**迭代收敛**. 否则，称此**迭代发散**.

4.1.2 迭代法敛散性的判别及其 MATLAB 程序

(一) 迭代法敛散性的判别

为了讨论由迭代公式(4.7)产生的迭代向量序列 $\{X^{(k)}\}$ 的敛散性，引进误差向量

$$R^{(k)}=X^{(k)}-X^*\quad(k=1,2,\cdots).$$

由(4.7)式减去(4.6)式，$R^{(k+1)}=BR^{(k)}\ (k=1,2,\cdots)$，递推得 $R^{(k)}=B^kR^{(0)}$. 要讨论 $\{X^{(k)}\}$ 是否收敛于 X^* 等价于研究 B 满足什么条件时，$R^{(k)}=B^kR^{(0)}\to 0$ $(k\to\infty)$，而 $B^k\to 0\ (k\to\infty)$ 又等价于 B 的所有特征根(取模)小于 1.

定义 4.1 若 n 阶矩阵 B 的特征根为 $\lambda_i\ (i=1,2,\cdots,n)$，称 $\rho(B)=\max\limits_{1\le i\le n}\{\lambda_i\}$ 为 B 的**谱半径**.

由上述分析可得.

定理 4.1 迭代公式(4.7)收敛的充要条件是谱半径 $\rho(\boldsymbol{B})<1$.

又因为矩阵的谱半径不超过它的任一种范数,所以迭代公式(4.7)收敛性还可以用下面的定理判别.

定理 4.2 若 $\|\boldsymbol{B}\|<1$,则对于任意初始向量 $\boldsymbol{X}^{(0)}$,迭代公式(4.7)都收敛,且有

$$\|\boldsymbol{X}^{(k)}-\boldsymbol{X}^{*}\| \leqslant \frac{\|\boldsymbol{B}\|}{1-\|\boldsymbol{B}\|}\|\boldsymbol{X}^{(k)}-\boldsymbol{X}^{(k-1)}\|\text{,或}$$

$$\|\boldsymbol{X}^{(k)}-\boldsymbol{X}^{*}\| \leqslant \frac{\|\boldsymbol{B}\|^{k}}{1-\|\boldsymbol{B}\|}\|\boldsymbol{X}^{(1)}-\boldsymbol{X}^{(0)}\|,$$

可见 $\|\boldsymbol{B}\|$ 越小收敛越快.

实际上,条件 $\rho(\boldsymbol{B})<1$, $\|\boldsymbol{B}\|<1$ 只能在构造出 $\boldsymbol{B}$ 以后来检验,我们更需要直接根据 $\boldsymbol{A}$ 来判断迭代的收敛性.

(二) 判别迭代法敛散性的 MATLAB 程序

根据定理 4.1 和谱半径定义,现提供一个名为 pddpb.m 的 M 文件,用于判别迭代公式(4.7)产生的迭代序列的敛散性.

> **用谱半径判别迭代法产生的迭代序列的敛散性的 MATLAB 主程序**
> 输入的量:线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的迭代公式(4.7)中的 $\boldsymbol{B}$;
> 输出的量:矩阵 $\boldsymbol{B}$ 的所有特征值和谱半径 $mH=\rho(\boldsymbol{B})$ 及其有关迭代法产生的迭代序列的敛散性的相关信息.

```
function H = ddpbj(B)
    H = eig(B);mH = norm(H,inf);
if mH > =1
    disp('请注意:因为谱半径不小于 1,所以迭代序列发散,谱半径 mH 和 B 的所有的特征值 H 如下:')
else
    disp('请注意:因为谱半径小于 1,所以迭代序列收敛,谱半径 mH 和 B 的所有的特征值 H 如下:')
end
mH
```

4.1.3 与迭代法有关的 MATLAB 命令

(一) 提取(产生)对角矩阵和特征值

可以用表 4-1 的 MATLAB 命令提取对角矩阵和特征值.

表 4－1 提取(产生)对角矩阵和特征值

MATLAB 命令	功 能
DX = diag(X)	若输入向量 X,则输出 DX 是以 X 为对角元的对角矩阵;若输入矩阵 X,则输出 DX 是以 X 的对角元构成的向量;
DX = diag(diag(X))	输入矩阵 X,输出 DX 是以 X 的对角元构成的对角矩阵,可用于迭代法中从 A 中提取 D.
lm = eig(A)	输入矩阵 A,输出 lm 是 A 的所有特征值.

例 4.1.3 已知向量 $X=(1,2,3,4)$ 和矩阵 $A=\begin{pmatrix}1&2&3&4\\1&-2&-3&-4\\0.1&0.2&0.3&0.4\\5&7&8&9\end{pmatrix}$,

求(1) 以 X 为对角元的对角矩阵 DX;

(2) 以 A 的对角元构成的向量 Y;

(3) 以 A 的对角元构成的对角矩阵 D;

(4) A 的所有特征值 lm.

解 (1)在 MATLAB 工作窗口输入程序

```
>> X=[1 2 3 4]; DX=diag(X)
```

运行后输出以 X 为对角元的对角矩阵 DX

```
DX =1    0    0    0
    0    2    0    0
    0    0    3    0
    0    0    0    4
```

(2) 在 MATLAB 工作窗口输入程序

```
>>A=[1 2 3 4;1 -2 -3 -4;0.1 0.2 0.3 0.4;5 7 8 9]; Y=diag(A)
```

运行后输出以 A 的对角元构成的向量为

```
Y=(1.000 0    -2.000 0    0.300 0    9.000 0)T
```

(3)在 MATLAB 工作窗口输入程序

```
>> A=[1 2 3 4;1 -2 -3 -4;0.1 0.2 0.3 0.4;5 7 8 9]; D=diag(diag(A))
```

运行后输出以 A 的对角元构成的对角矩阵 D 为

```
D =

    1.0000         0         0         0
         0   -2.0000         0         0
         0         0    0.3000         0
         0         0         0    9.0000
```

(4) 在 MATLAB 工作窗口输入程序

```
>> A=[1 2 3 4;1 -2 -3 -4;0.1 0.2 0.3 0.4;5 7 8 9]; lm=eig(A)
```

运行后输出 $\boldsymbol{A}$ 的所有特征值 lm

```
lm=(9.141 0   -0.420 5+1.418 1i   -0.420 5-1.418 1i   0.000 0)^T
```

(二) 提取(产生)上(下)三角形矩阵

可以用表 4-2 的 MATLAB 命令提取矩阵的上三角形矩阵和下三角形矩阵.

表 4-2 提取矩阵的上三角形矩阵和下三角形矩阵

MATLAB 命令	功 能
U=triu(A)	输入矩阵 $\boldsymbol{A}$,输出 $\boldsymbol{U}$ 是 $\boldsymbol{A}$ 的上三角形矩阵.
L=tril(A)	输入矩阵 $\boldsymbol{A}$,输出 $\boldsymbol{L}$ 是 $\boldsymbol{A}$ 的下三角形矩阵.
U=triu(A,1)	输入矩阵 $\boldsymbol{A}$,输出 $\boldsymbol{U}$ 是 $\boldsymbol{A}$ 的上三角形矩阵,但对角元为0,可用于迭代法中从 $\boldsymbol{A}$ 中提取 $\boldsymbol{U}$.
L=tril(A,-1)	输入矩阵 $\boldsymbol{A}$,输出 $\boldsymbol{L}$ 是 $\boldsymbol{A}$ 的下三角形矩阵,但对角元为0,可用于迭代法中从 $\boldsymbol{A}$ 中提取 $\boldsymbol{L}$.

例 4.1.4 从例 4.1.3 的矩阵 $\boldsymbol{A}$ 中提取如下矩阵:

(1) $\boldsymbol{A}$ 的上三角形矩阵 $\boldsymbol{U}$;(2) $\boldsymbol{A}$ 的下三角形矩阵 $\boldsymbol{L}$;(3) $\boldsymbol{A}$ 的上三角形矩阵 $\boldsymbol{U}$,但对角元为 0;(4) $\boldsymbol{A}$ 的下三角形矩阵 $\boldsymbol{L}$,但对角元为 0;(5) 考察命令 U2 =triu(A.2)和 L2 =tril(A,-2)的功能.

解 (1) 在 MATLAB 工作窗口输入程序

```
>> A=[1 2 3 4;1 -2 -3 -4;0.1 0.2 0.3 0.4;5 7 8 9]; U=triu(A)
```

运行后输出 $\boldsymbol{A}$ 的上三角形矩阵 $\boldsymbol{U}$

```
U =   1.0000     2.0000     3.0000     4.0000
           0    -2.0000    -3.0000    -4.0000
           0          0     0.3000     0.4000
           0          0          0     9.0000
```

(2) 在 MATLAB 工作窗口输入程序

```
>> A=[1 2 3 4;1 -2 -3 -4;0.1 0.2 0.3 0.4;5 7 8 9]; L=tril(A)
```

运行后输出 $\boldsymbol{A}$ 的下三角形矩阵 $\boldsymbol{L}$

```
L =   1.0000          0          0          0
      1.0000    -2.0000          0          0
      0.1000     0.2000     0.3000          0
      5.0000     7.0000     8.0000     9.0000
```

(3) 在 MATLAB 工作窗口输入程序:

```
>> A=[1 2 3 4;1 -2 -3 -4;0.1 0.2 0.3 0.4;5 7 8 9]; U=triu(A,1)
```

运行后输出 A 的上三角形矩阵 U,但对角元为 0

```
U =   0      2.0000      3.0000      4.0000
      0           0     -3.0000     -4.0000
      0           0           0      0.4000
      0           0           0           0
```

(4) 在 MATLAB 工作窗口输入程序

```
>> A=[1 2 3 4;1 -2 -3 -4;0.1 0.2 0.3 0.4;5 7 8 9]; L=tril(A,-1)
```

运行后输出 A 的下三角形矩阵 L,但对角元为 0

```
L =        0           0           0           0
      1.0000           0           0           0
      0.1000      0.2000           0           0
      5.0000      7.0000      8.0000           0
```

(5) 在 MATLAB 工作窗口输入程序

```
>> A=[1 2 3 4;1 -2 -3 -4;0.1 0.2 0.3 0.4;5 7 8 9];
   U2 = triu(A,2),L2 = tril(A,-2)
```

运行后输出结果

```
L2 =        0           0           0           0
            0           0           0           0
       0.1000           0           0           0
       5.0000      7.0000           0           0

U2 =   0     0     3     4
       0     0     0    -4
       0     0     0     0
       0     0     0     0
```

(三) 稀疏矩阵的处理

对稀疏矩阵在存贮和运算上的特殊处理,是 MATLAB 进行大规模科学计算时的特点和优势之一. 可以用表 4-3 的 MATLAB 命令,输入稀疏矩阵的非零元(零元不必输入),即可进行运算.

表 4-3 稀疏矩阵的 MATLAB 命令

MATLAB 命令	功 能
ZA=sparse(r,c,v,m,n)	表示在第 r 行、第 c 列输入数值 v,矩阵共 m 行 n 列,输出 ZA,给出 (r, c) 及 v 为一稀疏矩阵.
MA=full(ZA)	输入稀疏矩阵 ZA,输出为满矩阵 MA(包含零元)

例 4.1.5 首先将矩阵 $\boldsymbol{A}$ 用稀疏矩阵 $\boldsymbol{ZA}$ 表示，然后将 $\boldsymbol{ZA}$ 表示为满矩阵 $\boldsymbol{MA}$，其中

$$\boldsymbol{A}=\begin{pmatrix} 7 & & -1 & & \\ -4 & 7 & & -1 & \\ & -4 & 7 & & -1 \\ & & -4 & 7 & \\ & & & -4 & 7 \end{pmatrix}$$

解 输入 MATLAB 程序

```
>> ZA1 = sparse(1:5,1:5,7,5,5);
ZA2 = sparse(2:5,1:4, -4,5,5);
ZA3 = sparse(1:3, 3:5, -1,5,5);
ZA = ZA1 + ZA2 + ZA3,MA = full(ZA)
```

运行后输出结果

```
ZA =
   (1,1)      7              MA =    7     0    -1     0     0
   (2,1)     -4                     -4     7     0    -1     0
   (2,2)      7                      0    -4     7     0    -1
   (3,2)     -4                      0     0    -4     7     0
   (1,3)     -1                      0     0     0    -4     7
   (3,3)      7
   (4,3)     -4
   (2,4)     -1
   (4,4)      7
   (5,4)     -4
   (3,5)     -1
   (5,5)      7
```

用下面的例子说明稀疏矩阵处理的优点.

例 4.1.6 用稀疏矩阵和满矩阵分别求解 $\boldsymbol{AX}=\boldsymbol{b}$，进行比较，其中 $n=2\,000$，

$$\boldsymbol{A}=\begin{pmatrix} 5 & 2 & & & & \\ 3 & 5 & 2 & & & \\ & 3 & 5 & 2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 3 & 5 & 2 \\ & & & & 3 & 5 \end{pmatrix}_{n\times n},\boldsymbol{b}=\begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ n-1 \\ n \end{pmatrix}$$

解 输入 MATLAB 程序

```
>> n=2000; b=[1:n]'; a1=sparse(1:n,1:n,5,n,n);
a2=sparse(2:n,1:n-1,3,n,n); a3=sparse(2:n,1:n-1,2,n,n);
a=a1+a2+a3';
tic; x=a\b; t1=toc% 输出用稀疏矩阵求解的时间 t1
aa=full(a); % 与满矩阵作比较
tic; xx=aa\b; t2=toc% 输出用满矩阵求解的时间 t2
y=sum(x),yy=sum(xx)% 为检验 x 与 xx 相同分别输出其分量之和
```

运行后输出结果

```
t1=0, t2=2.6900, y=2.0020e+005, yy=2.0020e+005
```

由此可见,结果 y 与 yy 相同,而 t_1 与 t_2 相差很大.



习 题 4.1

1. 已知向量 $\boldsymbol{X}=(-1,1,2,3,4)$ 和矩阵 $\boldsymbol{A}=\begin{pmatrix}70&2&3&4\\1&-2&-3&-4\\0.1&0.2&0.3&0.4\\5&7&8&9\end{pmatrix}$, 求(1) 以 $\boldsymbol{X}$ 为对角元的对角矩阵 $\boldsymbol{DX}$;(2) 以 $\boldsymbol{A}$ 的对角元构成的向量 $\boldsymbol{Y}$;(3) 以 $\boldsymbol{A}$ 的对角元构成的对角矩阵 $\boldsymbol{D}$;(4) $\boldsymbol{A}$ 的所有特征值 lm.

2. 从矩阵 $\boldsymbol{A}=\begin{pmatrix}15&2&3&4\\10&-2&-3&-4\\0.1&0.2&0.3&0.4\\5&7&8&9\end{pmatrix}$ 中提取如下矩阵:

(1) $\boldsymbol{A}$ 的上三角形矩阵 $\boldsymbol{UA}$;(2) $\boldsymbol{A}$ 的下三角形矩阵 $\boldsymbol{LA}$;(3) $\boldsymbol{A}$ 的上三角形矩阵 $\boldsymbol{U}$,但对角元为 0;(4) $\boldsymbol{A}$ 的下三角形矩阵 $\boldsymbol{L}$,但对角元为 0;(5) 考察命令 `U2 = triu(A,2)` 和 `L2 = tril(A,-2)` 的功能.

3. 首先将矩阵 $\boldsymbol{A}$ 用稀疏矩阵 $\boldsymbol{ZA}$ 表示,然后将 $\boldsymbol{ZA}$ 表示为满矩阵 $\boldsymbol{MA}$,其中

$$\boldsymbol{A}=\begin{pmatrix}6&&-1&&\\-2&6&&-1&\\&-2&6&&-1\\&&-2&6&\\&&&-2&6\end{pmatrix}$$

4. 用稀疏矩阵和满矩阵分别求解 $\boldsymbol{AX}=\boldsymbol{b}$,进行比较,其中 $n=20\,000$,

$$A=\begin{pmatrix}15 & 22 & & & & \\ 43 & 15 & 22 & & & \\ & 43 & 15 & 22 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 43 & 15 & 22 \\ & & & & 43 & 15\end{pmatrix}_{n\times n},\quad \boldsymbol{b}=\begin{pmatrix}1\\2\\\vdots\\n\end{pmatrix}$$

5. 通过试验观察 MATLAB 软件处理稀疏矩阵的优势，求解 $\boldsymbol{AX}=\boldsymbol{b}$，其中

$$\boldsymbol{A}=\begin{pmatrix}4 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 4\end{pmatrix}_{n\times n}$$

，$\boldsymbol{b}$ 是任意 n 维列向量（n 在 100 以上），用稀疏矩阵和满矩阵两种算法计算，比较所需时间.

4.2 雅可比(Jacobi)迭代及其 MATLAB 程序

4.2.1 雅可比迭代

如果线性方程组

$$\begin{cases}a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=b_1,\\ a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n=b_2,\\ \cdots\cdots\cdots\cdots\\ a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n=b_n,\end{cases}\tag{4.8}$$

的系数 $a_{ii}\neq 0\quad(i=1,2,\cdots,n)$，则线性方程组(4.8)的**雅可比迭代公式**为

$$\begin{cases}x_1^{(k+1)}=\dfrac{1}{a_{11}}(-a_{12}x_2^{(k)}-a_{13}x_3^{(k)}-\cdots-a_{1n}x_n^{(k)}+b_1),\\ x_2^{(k+1)}=\dfrac{1}{a_{22}}(-a_{21}x_1^{(k)}-a_{23}x_3^{(k)}-\cdots-a_{2n}x_n^{(k)}+b_2),\\ \cdots\cdots\cdots\cdots\\ x_n^{(k+1)}=\dfrac{1}{a_{nn}}(-a_{n1}x_1^{(k)}-a_{n2}x_2^{(k)}-\cdots-a_{n,n-1}x_{n-1}^{(k)}+b_n).\end{cases}\tag{4.9}$$

将(4.8)式的系数矩阵 $\boldsymbol{A}$ 分解为 $\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$，其中

$$D = \mathrm{diag}(a_{11}, a_{22}, \cdots, a_{nn}) = \begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix},$$

$$L = -\begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & a_{n2} & \cdots\ a_{n,n-1} & 0 \end{pmatrix}, U = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & 0 & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{pmatrix}.$$

若对角矩阵 D 非奇异(即 $a_{ii} \neq 0, i=1,2,\cdots,n$),则(4.9)式化为

$$X = D^{-1}(L + U)X + D^{-1}b. \tag{4.10}$$

不难看出,例 4.1.1 和例 4.1.2 中(4.3)式和(4.4)式正是这种形式. 若记

$$B_1 = D^{-1}(L + U), f_1 = D^{-1}b, \tag{4.11}$$

则方程组(4.10)的迭代形式可写作

$$X^{(k+1)} = B_1X^{(k)} + f_1 \qquad (k = 0,1,2\cdots). \tag{4.12}$$

如果由(4.12)式产生的迭代向量序列 $\{X^{(k)}\}$ 收敛于 X^*,则 X^* 必是方程(4.10)式的解,因而也是(4.5)式的解. (4.11)式、(4.12)式称**雅可比迭代**.

例 4.2.1 试用雅可比迭代法解线性方程组

$$\begin{cases} 10x_1 + 2x_2 + 3x_3 = 1, \\ 2x_1 + 10x_2 + x_3 = 1, \\ 3x_1 + x_2 + 10x_3 = 2, \end{cases}$$

取初始值 $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0,0,0)$,要求当 $\| x^{(k+1)} - x^{(k)} \|_\infty < 10^{-3}$ 时,迭代终止.

解 雅可比迭代公式为

$$\begin{cases} x_1^{(k+1)} = \dfrac{1}{10}(1 - 2x_2^{(k)} - 3x_3^{(k)}), \\ x_2^{(k+1)} = \dfrac{1}{10}(1 - 2x_1^{(k)} - x_3^{(k)}), \\ x_3^{(k+1)} = \dfrac{1}{10}(2 - 3x_1^{(k)} - x_2^{(k)}), \end{cases} \qquad k = 0,1,2,\cdots. \tag{4.13}$$

取初始值 $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0,0,0)$,计算结果如表 4-4. 当 $\| x^{(7)} - x^{(6)} \|_\infty < 10^{-3}$ 时,迭代终止,线性方程组的近似解为 $x_1 \approx 0.030, x_2 \approx 0.076, x_3 \approx 0.184$.

表 4-4

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	0	0	0
1	0.100 0	0.100 0	0.200 0
2	0.020 0	0.060 0	0.160 0
3	0.040 0	0.080 0	0.188 0
4	0.027 6	0.073 2	0.180 0
5	0.031 4	0.076 5	0.184 4
6	0.029 4	0.075 3	0.182 9
7	0.030 1	0.075 8	0.183 7

4.2.2 雅可比迭代的收敛性及其 MATLAB 程序

(一) 雅可比迭代的收敛性

我们可以根据定理 4.1 和定理 4.2 判别由雅可比迭代公式(4.12)产生的迭代向量序列$\{X^{(k)}\}$是否收敛于原方程组的解 X^*,但是这两个定理的条件 $\rho(B)<1$, $\|B\|<1$ 只能在构造出 B 以后来检验,我们更需要直接根据 A 来判断迭代的收敛性,为此介绍严格对角占优矩阵的概念.

定义 4.2 如果 $n\times n$ 阶矩阵 $A=(a_{ij})_{n\times n}$的元满足

$$|a_{kk}|>\sum_{\substack{j=1\\j\neq k}}^{n}|a_{kj}|\quad(k=1,2,\cdots,n),\tag{4.14}$$

则称矩阵 A 是严格对角占优的.

定理 4.3 如果 $n\times n$ 阶矩阵 A 是严格对角占优的,则线性方程组(4.5)有唯一解 X^*,且对于任意初始向量 $X^{(0)}$,由(4.11)式、(4.12)式产生的迭代向量序列$\{X^{(k)}\}$都收敛于 X^*.

(二) 判别雅可比迭代收敛性的 MATLAB 程序

根据定理 4.3 和公式(4.14),现提供一个名为 jspb.m 的 M 文件如下:

判别雅可比迭代收敛性的 MATLAB 主程序

输入的量:线性方程组 $AX=b$ 的系数矩阵 A;

输出的量:系数矩阵 $A=(a_{ij})_{n\times n}$的 $a=\sum_{\substack{j=1\\j\neq k}}^{n}|a_{kj}|-|a_{kk}|\quad(k=1,2,\cdots,n)$ 的值和有关雅可比迭代收敛性的相关信息.

```
function a = jspb(A)
[n m] = size(A);
  for j = 1:m
         a(j) = sum(abs(A(:,j))) -2 * (abs(A(j,j)));
  end
  for i = 1:n
     if a(i) > =0
            disp('请注意:系数矩阵 A 不是严格对角占优的,此雅可比迭代不一定收敛')
            return
     end
  end
if a(i) <0
         disp('请注意:系数矩阵 A 是严格对角占优的,此方程组有唯一解,且雅可比迭代收敛 ')
end
```

例 4.2.2 用判别雅可比迭代收敛性的 MATLAB 主程序,判别由下列方程组的雅可比迭代产生的序列是否收敛?

(1) $\begin{cases} 10x_1 - x_2 - 2x_3 = 7.2, \\ -x_1 + 10x_2 - 2x_3 = 8.3, \\ -x_1 - x_2 + 5x_3 = 4.2; \end{cases}$ (2) $\begin{cases} 10x_1 - x_2 - 2x_3 = 7.2, \\ -x_1 + 10x_2 - 2x_3 = 8.3, \\ -x_1 - x_2 + 0.5x_3 = 4.2. \end{cases}$

解 (1) 首先保存名为 jspb.m 的 M 文件,然后在 MATLAB 工作窗口输入程序

```
>> A = [10  -1  -2; -1 10  -2; -1  -1 5];a = jspb(A)
```

运行后输出结果

```
请注意:系数矩阵 A 是严格对角占优的,此方程组有唯一解,且雅可比迭代收敛
a =
    -8  -8  -1
```

(2) 在 MATLAB 工作窗口输入程序

```
>> A = [10  -1  -2; -1 10  -2; -1  -1 0.5];a = jspb(A)
```

运行后输出结果

```
请注意:系数矩阵 A 不是严格对角占优的,此雅可比迭代不一定收敛
a =
    -8.0000e+000  -8.0000e+000 3.5000e+000
```

4.2.3 雅可比迭代的两种 MATLAB 程序

利用 MATLAB 程序和雅可比迭代解线性方程组 $AX=b$ 的常用的方法有两种,一种方法是根据雅可比迭代公式(4.11)、(4.12)式、定理 4.3 和公式(4.14)编写一个名为 jacdd.m 的 M 文件并保存,然后在 MATLAB 工作窗口输入对应的命令,执行此文件;另一种方法是根据雅可比迭代的定义,利用提取对角矩阵和上、下三角形矩阵的 MATLAB 程序解线性方程组 $AX=b$. 下面我们分别介绍这两种方法.

(一) 雅可比迭代公式的 MATLAB 程序

用雅可比迭代解线性方程组 $AX=b$ 的 MATLAB 主程序

输入的量:线性方程组 $AX=b$ 的系数矩阵 A 和 b, 初始向量 X_0,范数的名称 $P=1,2,$ inf 或 'fro.',近似解 X 的误差(精度)*wucha* 和迭代的最大次数 *max*1;

输出的量:系数矩阵 $A=(a_{ij})_{n\times n}$的 $a=\sum\limits_{\substack{j=1\\ i\neq k}}^{n}|a_{kj}|-|a_{kk}|\quad(k=1,2,\cdots,n)$ 的值和有关雅可比迭代收敛性的相关信息及其 $AX=b$ 的精确解 jX 和近似解 X.

根据雅可比迭代(4.11)式、(4.12)式、定理 4.3 和公式(4.14),现提供一个名为 jacdd.m 的 M 文件如下:

```
function X = jacdd(A,b,X0,P,wucha,max1)
[n m] = size(A);
   for j = 1:m
      a(j) = sum(abs(A(:,j))) - 2 * (abs(A(j,j)));
   end
   for i = 1:n
      if a(i) > = 0
         disp('请注意:系数矩阵 A 不是严格对角占优的,此雅可比迭代不一定收敛')
         return
      end
    end
 if a(i) < 0
         disp('请注意:系数矩阵 A 是严格对角占优的,此方程组有唯一解,且雅可比迭代收敛 ')
```

```
end
for k =1:max1
       k
  for j =1:m
    X(j) =(b(j) -A(j,[1:j-1,j +1:m]) * X0([1: j -1,j +1:m])) /A
    (j,j);
  end
  X,djwcX =norm(X' -X0,P); xdwcX =djwcX /(norm(X',P) +eps); X0
   =X';X1 =A \b;
  if (djwcX <wucha)&(xdwcX <wucha)
     disp('请注意:雅可比迭代收敛,此方程组的精确解 jX 和近似解 X 如下:')
      return
  end
end
if(djwcX >wucha)&(xdwcX >wucha)
   disp('请注意:雅可比迭代次数已经超过最大迭代次数 max1 ')
end
a,X =X;jX =X1',
```

例 4.2.3 用∞范数和判别雅可比迭代的 MATLAB 主程序解例 4.2.2 中的方程组,解的精度为 0.001,分别取最大迭代次数 $max1 = 100, 5$,初始向量 $X_0 = (0\ 0\ 0)^T$,并比较它们的收敛速度.

解 (1) 取最大迭代次数 $max1 = 100$ 时.

① 首先保存名为 jacdd.m 的 M 文件,然后在 MATLAB 工作窗口输入程序

```
>> A =[10 -1 -2; -1 10 -2; -1 -1 5]; b =[7.2;8.3;4.2];
X0 =[0 0 0]'; X =jacdd(A,b,X0,inf,0.001,100)
```

运行后输出结果

```
请注意:系数矩阵 A 是严格对角占优的,此方程组有唯一解,且雅可比迭代收敛
请注意:雅可比迭代收敛,此方程组的精确解 jX 和近似解 X 如下:
a =
   -8     -8     -1
jX =
    1.1000    1.2000    1.3000
X =
   1.0994    1.1994    1.2993
```

② 在 MATLAB 工作窗口输入程序

```
>> A =[10 -1 -2; -1 10 -2; -1 -1 0.5]; b =[7.2;8.3;4.2];
X0 =[0 0 0]';
```

```
X = jacdd(A,b,X0,inf, 0.001,100)
```

运行后输出结果

```
请注意:系数矩阵 A 不是严格对角占优的,此雅可比迭代不一定收敛
请注意:雅可比迭代收敛,此方程组的精确解 jX 和近似解 X 如下:
a =
   -8.0000     -8.0000     3.5000
jX =
    24.5000     24.6000     106.6000
X =
   24.0738     24.1738     104.7974
```

(2) 取最大迭代次数 $max1 = 5$ 时,

① 在 MATLAB 工作窗口输入程序

```
>> A = [10 -1  2; -1 10 -2; -1 -1 5];
 b = [7.2;8.3;4.2]; X0 = [0 0 0]'; X = jacdd(A,b,X0,inf,0.001,5)
```

运行后输出结果

```
请注意:系数矩阵 A 是严格对角占优的,此方程组有唯一解,雅可比迭代收敛
请注意:雅可比迭代次数已经超过最大迭代次数 max1
a =
   -8     -8     -1
jX =
    1.1000     1.2000     1.3000
X =
   1.0951     1.1951     1.2941
```

② 在 MATLAB 工作窗口输入程序

```
>> A = [10 -1 -2; -1 10 -2; -1 -1 0.5]; b = [7.2;8.3;4.2];
 X0 = [0 0 0]'; X = jacdd(A,b,X0,inf, 0.001,5)
```

运行后输出结果

```
请注意:系数矩阵 A 不是严格对角占优的,此雅可比迭代不一定收敛
请注意:雅可比迭代次数已经超过最大迭代次数 max1
a =
   -8.0000     -8.0000     3.5000
jX =
    24.5000     24.6000     106.6000
X =
   5.5490     5.6490     27.6553
```

由(1)和(2)可见,如果系数矩阵 $\boldsymbol{A}$ 是严格对角占优的,则雅可比迭代收敛的速度快;如果系数矩阵 $\boldsymbol{A}$ 不是严格对角占优的,则雅可比迭代可能收敛,也可

能发散. 如果收敛,则收敛的速度慢. 因此,当 $a = \sum_{\substack{j=1 \\ i \neq k}}^{n} |a_{kj}| - |a_{kk}| < 0 \quad (k = 1, 2, \cdots, n)$ 时, $|a|$ 的值越小,雅可比迭代收敛的速度越快.

(二) 雅可比迭代定义的 MATLAB 程序

利用雅可比迭代定义编写解线性方程组(4.5)的 MATLAB 程序的一般步骤

步骤 1 将线性方程组(4.5)的系数矩阵 $\boldsymbol{A}$ 分解为 $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}$,其中

$$\boldsymbol{D} = \mathrm{diag}(a_{11}, a_{22}, \cdots a_{nn}) = \begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix},$$

$$\boldsymbol{L} = -\begin{pmatrix} 0 & & & & \\ a_{21} & 0 & & & \\ \vdots & \ddots & \ddots & & \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \boldsymbol{U} = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & 0 & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{pmatrix}$$

在 MATLAB 工作窗口输入程序

```
>>A=[a11 a12 …a1n; a21 a22 …a2n;…; an1 an2 …ann;];
  D=diag(A)U=triu(A,1), L=tril(A,-1)
```

运行后即可输出 $\boldsymbol{A}$ 的 $\boldsymbol{D}, \boldsymbol{L}, \boldsymbol{U}$;

步骤 2 若对角矩阵 $\boldsymbol{D}$ 非奇异(即 $a_{ii} \neq 0, i = 1, \cdots, n$),则(4.5)式化为

$$\boldsymbol{X} = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})\boldsymbol{X} + \boldsymbol{D}^{-1}\boldsymbol{b}.$$

若记

$$\boldsymbol{B}_1 = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U}), \boldsymbol{f}_1 = \boldsymbol{D}^{-1}\boldsymbol{b}.$$

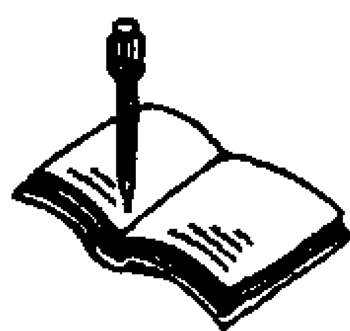
则方程组的迭代形式可写作

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{B}_1 \boldsymbol{X}^{(k)} + \boldsymbol{f}_1 \quad (k = 0, 1, 2 \cdots)$$

可以利用下面的 MATLAB 程序完成

```
>>dD=det(D);
if dD==0
   disp('请注意:因为对角矩阵 D 奇异,所以此方程组无解.')
else
  disp('请注意:因为对角矩阵 D 非奇异,所以此方程组有解.')
  iD=inv(D); B1=iD*(L+U);f1=iD*b;
  for k=1:max1
      X = B1*X0 + f1; X0=X; djwcX=norm(X-X0,P);
        xdwcX=djwcX/(norm(X,P)+eps); X1=A\b;
```

```
        if(djwcX < wucha)&(xdwcX < wucha)
            disp('请注意:雅可比迭代收敛,此方程组的精确解 jX 和近似解 X
             如下:')
             return
        end
    end
    if (djwcX > wucha)|(xdwcX > wucha)
       disp('请注意:雅可比迭代次数已经超过最大迭代次数 max1 ')
    end
end
a,X = X;jX = X1 ',
```



习 题 4.2

1. 用判别雅可比迭代收敛性的 MATLAB 主程序,判别由下列方程组的雅可比迭代产生的序列是否收敛?

(1) $\begin{cases} 23x_1 - x_2 - 2x_3 = 1.7, \\ -x_1 + 10x_2 - 2x_3 = 8.3, \\ -x_1 - x_2 + 5x_3 = 4.2; \end{cases}$ (2) $\begin{cases} 15x_1 - x_2 - 2x_3 = 7.2, \\ -x_1 + 10x_2 - 2x_3 = 8.3, \\ -x_1 - x_2 + 0.5x_3 = 4.2; \end{cases}$

(3) $\begin{cases} 10x_1 - x_2 - 2x_3 = 7.2, \\ -x_1 + 10x_2 - 2x_3 = 8.3, \\ -x_1 - x_2 + 5x_3 = 4.2; \end{cases}$ (4) $\begin{cases} x_1 + 2x_2 + 3x_3 = 14, \\ 2x_1 + 5x_2 + 2x_3 = 18, \\ 3x_1 + x_2 + 5x_3 = 20. \end{cases}$

2. 分别用∞范数和判别雅可比迭代的 MATLAB 主程序解第 1 题中的方程组,解的精度为 0.001,分别取最大迭代次数 $max1 = 100, 5$,初始向量 $\boldsymbol{X}_0 = (0\ 0\ 0)^T$,并比较它们的收敛速度.

3. 设有线性方程组 $\begin{cases} x_1 + 2x_2 + 3x_3 = 14, \\ 2x_1 + 5x_2 + 2x_3 = 18, \\ 3x_1 + x_2 + 5x_3 = 20. \end{cases}$

(1) 用高斯消元法解该方程组;

(2) 写出用雅可比迭代法解该方程组的迭代公式.

4. 试用雅可比迭代法解线性方程组 $\begin{cases} 10x_1 + 2x_2 + 3x_3 = 14, \\ 2x_1 + 10x_2 + x_3 = 11, \\ 3x_1 + x_2 + 10x_3 = 20. \end{cases}$ 取初始值 $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0,0,0)$,要求当 $\| \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \|_\infty < 10^{-3}$ 时,迭代终止.

4.3 高斯-塞德尔(Gauss-Seidel)迭代及其 MATLAB 程序

4.3.1 高斯-塞德尔迭代

仔细研究一下雅可比迭代公式(4.9)式可以发现,它能够改进为

$$\begin{cases} x_1^{(k+1)} = \dfrac{1}{a_{11}}(-a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)} + b_1), \\ x_2^{(k+1)} = \dfrac{1}{a_{22}}(-a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)} + b_2), \\ \cdots\cdots\cdots\cdots \\ x_n^{(k+1)} = \dfrac{1}{a_{nn}}(-a_{n1}x_1^{(k+1)} - a_{n2}x_2^{(k+1)} - \cdots - a_{n,n-1}x_{n-1}^{(k+1)} + b_n). \end{cases} \tag{4.15}$$

这就是说在雅可比迭代过程中,计算 $x_i^{(k+1)}$ 时 $x_1^{(k+1)},\cdots,x_{i-1}^{(k+1)}$ 已经得到,不必再用 $x_1^{(k)},\cdots,x_{i-1}^{(k)}$,即原来的迭代公式 $\boldsymbol{D}\boldsymbol{X}^{(k+1)}=\boldsymbol{L}\boldsymbol{X}^{(k)}+\boldsymbol{U}\boldsymbol{X}^{(k)}+\boldsymbol{b}$ 可以改进为 $\boldsymbol{D}\boldsymbol{X}^{(k+1)}=\boldsymbol{L}\boldsymbol{X}^{(k+1)}+\boldsymbol{U}\boldsymbol{X}^{(k)}+\boldsymbol{b}$. 在 $\boldsymbol{D}$ 非奇异的假设下,$(\boldsymbol{D}-\boldsymbol{L})$ 可逆,于是得到

$$\boldsymbol{B}_2 = (\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{U},\ \boldsymbol{f}_2 = (\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{b} \tag{4.16}$$

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{B}_2\boldsymbol{X}^{(k)} + \boldsymbol{f}_2 \quad (k=0,1,2\cdots), \tag{4.17}$$

称为**高斯-塞德尔迭代**. 它与雅可比迭代相比的优点,不仅在于用 $\boldsymbol{X}$ 的新分量代替旧分量精度会高些,而且在迭代过程中不必用两个工作单元存放 $\boldsymbol{X}^{(k)}$, $\boldsymbol{X}^{(k+1)}$,可以只用一套工作单元,即一个分量、一个分量地将 $\boldsymbol{X}^{(k)}$ 替换为 $\boldsymbol{X}^{(k+1)}$.

例 4.3.1 试用高斯-塞德尔迭代法解线性方程组 $\begin{cases} 10x_1+2x_2+3x_3=1, \\ 2x_1+10x_2+x_3=1, \\ 3x_1+x_2+10x_3=2, \end{cases}$ 取初始值 $(x_1^{(0)},x_2^{(0)},x_3^{(0)})=(0,0,0)$,要求当 $\|\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}\|_\infty<10^{-3}$ 时,迭代终止. 并将计算结果与例 4.2.1 比较.

解 高斯-塞德尔迭代公式为

$$\begin{cases} x_1^{(k+1)} = \dfrac{1}{10}(1-2x_2^{(k)}-3x_3^{(k)}), \\ x_2^{(k+1)} = \dfrac{1}{10}(1-2x_1^{(k+1)}-x_3^{(k)}), \qquad k=0,1,2,\cdots. \\ x_3^{(k+1)} = \dfrac{1}{10}(2-3x_1^{(k+1)}-x_2^{(k+1)}), \end{cases}$$

取初始值 $(x_1^{(0)},x_2^{(0)},x_3^{(0)})=(0,0,0)$,计算结果如下:

$$\begin{cases} x_1^{(1)} = \dfrac{1}{10}(1 - 2x_2^{(0)} - 3x_3^{(0)}) = 0.100\,0, \\ x_2^{(1)} = \dfrac{1}{10}(1 - 2x_1^{(1)} - x_3^{(0)}) = 0.080\,0, \\ x_3^{(1)} = \dfrac{1}{10}(2 - 3x_1^{(1)} - x_2^{(1)}) = 0.162\,0, \end{cases}$$

$$\begin{cases} x_1^{(2)} = \dfrac{1}{10}(1 - 2x_2^{(1)} - 3x_3^{(1)}) = 0.035\,4, \\ x_2^{(2)} = \dfrac{1}{10}(1 - 2x_1^{(2)} - x_3^{(1)}) = 0.076\,7, \\ x_3^{(2)} = \dfrac{1}{10}(2 - 3x_1^{(2)} - x_2^{(2)}) = 0.181\,7, \end{cases}$$

$$\begin{cases} x_1^{(3)} = \dfrac{1}{10}(1 - 2x_2^{(2)} - 3x_3^{(2)}) = 0.030\,1, \\ x_2^{(3)} = \dfrac{1}{10}(1 - 2x_1^{(3)} - x_3^{(2)}) = 0.075\,8, \\ x_3^{(3)} = \dfrac{1}{10}(2 - 3x_1^{(3)} - x_2^{(3)}) = 0.183\,4, \end{cases}$$

因为 $\| \boldsymbol{x}^{(3)} - \boldsymbol{x}^{(2)} \|_\infty < 10^{-3}$ 时，迭代终止，线性方程组的近似解为

$$x_1 \approx 0.030, x_2 \approx 0.076, x_3 \approx 0.183.$$

将计算结果与例 4.2.1 比较，雅可比迭代比高斯—塞德尔迭代收敛的速度慢.

4.3.2 高斯 - 塞德尔迭代的收敛性

我们可以根据定理 4.1、定理 4.2 和定理 4.3 判别由高斯 - 塞德尔迭代公式(4.16)、(4.17)产生的迭代向量序列 $\{\boldsymbol{X}^{(k)}\}$ 是否收敛于原方程组的解 $\boldsymbol{X}^*$. 另外，若 $\boldsymbol{A}$ 是对称正定矩阵，则也可以用下面定理判别 $\boldsymbol{AX} = \boldsymbol{b}$ 的高斯 - 塞德尔迭代收敛性.

定理 4.4 如果 $n \times n$ 阶矩阵 $\boldsymbol{A}$ 是正定的对称矩阵，则对于任意初始向量 $\boldsymbol{X}^{(0)}$，由高斯—塞德尔迭代(4.16)、(4.17)产生的迭代向量序列 $\{\boldsymbol{X}^{(k)}\}$ 都收敛于线性方程组(4.5)的解 $\boldsymbol{X}^*$.

例 4.3.2 用雅可比迭代和高斯 - 塞德尔迭代计算下列线性方程组，并且比较两种迭代的收敛速度.

$$\begin{cases} 10x_1 + 3x_2 + x_3 = 14, \\ 2x_1 - 10x_2 + 3x_3 = -5, \\ x_1 + 3x_2 + 10x_3 = 14. \end{cases}$$

取初始值 $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0,0,0)$，迭代 4 次终止. 并将两种方法的计算结果

与精确解$(x_1,x_2,x_3)=(1.188\,418\cdots,\ 1.029\,386\cdots,\ 0.972\,342\cdots)$进行比较.

解 原线性方程组的雅可比迭代公式为

$$\begin{cases} x_1^{(k+1)} = -0.3x_2^{(k)} - 0.1x_3^{(k)} + 1.4, \\ x_2^{(k+1)} = 0.2x_1^{(k)} + 0.3x_3^{(k)} + 0.5, \\ x_3^{(k+1)} = -0.1x_1^{(k)} - 0.3x_2^{(k)} + 1.4. \end{cases}$$

原线性方程组的高斯-塞德尔迭代公式为

$$\begin{cases} x_1^{(k+1)} = -0.3x_2^{(k)} - 0.1x_3^{(k)} + 1.4, \\ x_2^{(k+1)} = 0.2x_1^{(k+1)} + 0.3x_3^{(k)} + 0.5, \\ x_3^{(k+1)} = -0.1x_1^{(k+1)} - 0.3x_2^{(k+1)} + 1.4. \end{cases}$$

取初始迭代向量$X_0=(0\ 0\ 0)^T$,分别代入雅可比迭代公式和高斯-塞德尔迭代公式,计算结果列入表4-5,显然,高斯-塞德尔迭代比雅可比迭代收敛速度要快(见表4-5).

表4-5 用雅可比法和高斯-塞德尔法计算的结果

k	x^T(雅可比)	x^T(高斯-塞德尔)
0	(0, 0, 0)	(0, 0, 0)
1	(1.4, 0.5, 1.4)	(1.4, 0.78, 1.026)
2	(1.39, 1.2, 1.11)	(1.268 6; 1.061 5, 0.954 7)
3	(1.151, 1.111, 0.901)	(1.177 0, 1.021 8, 0.975 8)
4	(1.156 8, 1.000 5, 0.951 6)	(1.191 0, 1.030 9, 0.971 6)

4.3.3 高斯-塞德尔迭代的两种 MATLAB 程序

利用 MATLAB 程序和高斯-塞德尔迭代解线性方程组$AX=b$的常用方法有两种,一种方法是根据高斯-塞德尔迭代公式(4.16)、(4.17)、定理4.3和公式(4.14)编写一个名为 gsdd.m 的 M 文件并保存,然后在 MATLAB 工作窗口输入对应的命令,执行此文件;另一种方法是根据高斯-塞德尔迭代的定义,利用提取对角矩阵和上、下三角形矩阵的 MATLAB 程序解线性方程组$AX=b$. 下面我们分别介绍这两种方法.

(一) 高斯-塞德尔迭代定义的 MATLAB 程序

根据高斯-塞德尔迭代定义,现提供名为 gsdddy.m 的 M 文件如下:

用高斯 - 塞德尔迭代定义解线性方程组 $AX=b$ 的 MATLAB 主程序

输入的量：线性方程组 $AX=b$ 的系数矩阵 A 和 b，初始向量 X_0，范数的名称 $P=1, 2$, inf, 或 'fro.'，近似解 X 的误差（精度）*wucha* 和迭代的最大次数 *max*1.

输出的量：以系数矩阵 $A=(a_{ij})_{n\times n}$ 的对角元构成的对角矩阵 D、A 的上三角形矩阵 U，但对角元为 0、A 的下三角形矩阵 L，但对角元为 0 和有关高斯 - 塞德尔迭代收敛性的相关信息及其 $AX=b$ 的精确解 jX 和近似解 X.

```
function X = gsdddy(A,b,X0,P,wucha,max1)
  D = diag(diag(A));U = -triu(A,1);
  L = -tril(A, -1); dD = det(D);
if dD == 0
     disp('请注意:因为对角矩阵 D 奇异,所以此方程组无解.')
else
     disp('请注意:因为对角矩阵 D 非奇异,所以此方程组有解.')
     iD = inv(D - L); B2 = iD * U;f2 = iD * b;jX = A \b;
     X = X0; [n m] = size(A);
     for k = 1:max1
     X1 - B2 * X + f2; djwcX = norm(X1 - X,P);
     xdwcX = djwcX /(norm(X,P) + eps);
       if (djwcX < wucha) |(xdwcX < wucha)
             return
       else
              k,X1 ',k = k + 1;X = X1;
       end
      end
      if (djwcX < wucha) |(xdwcX < wucha)
             disp('请注意:高斯 - 塞德尔迭代收敛,此 A 的分解矩阵 D,U,L
和方程组的精确解 jX 和近似解 X 如下:')
      else
         disp('请注意:高斯 - 塞德尔迭代的结果没有达到给定的精度,并且
迭代次数已经超过最大迭代次数 max1,方程组的精确解 jX 和迭代向量 X 如下:')
         X = X';jX = jX'
      end
  end
   X - X';D,U,L,jX = jX'
```

例 4.3.3 用高斯 - 塞德尔迭代定义的 MATLAB 主程序解下列线性方程

组,取初始值$(x_1^{(0)},x_2^{(0)},x_3^{(0)})=(0,0,0)$,要求当$\|\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}\|_\infty<10^{-3}$时,迭代终止.

$$(1)\begin{cases}10x_1-x_2-2x_3=7.2,\\-x_1+10x_2-2x_3=8.3,\\-x_1-x_2+0.5x_3=4.2;\end{cases}\quad(2)\begin{cases}3x_1+4x_2-5x_3+7x_4=5,\\2x_1-8x_2+3x_3-2x_4=2,\\4x_1+51x_2-13x_3+16x_4=-1,\\7x_1-2x_2+21x_3+3x_4=21.\end{cases}$$

解 (1) 首先保存名为 gsdddy.m 的 M 文件,然后在 MATLAB 工作窗口输入程序

```
>> A=[10 -1 -2;-1 10 -2;-1 -1 0.5];
   b-[7.2;8.3;4.2]; X0=[0 0 0]';
   X=gsdddy(A,b,X0,inf, 0.001,100)
```

运行后输出结果

```
请注意:因为对角矩阵 D 非奇异,所以此方程组有解.
请注意:高斯－塞德尔迭代收敛,此 A 的分解矩阵 D,U,L 和方程组的精确解 jX
和近似解 X 如下:
D =                                    L =
   10.0000         0         0           0     0     0
         0   10.0000         0           1     0     0
         0         0    0.5000           1     1     0
U =                                    jX =
   0     1     2                           24.5000   24.6000   106.6000
   0     0     2                       X =
   0     0     0                          24.4996   24.5996   106.5984
```

此近似解与例 4.2.3 中的(1)中②的解 $\boldsymbol{X}=(24.0738, 24.1738, 104.7974)^{\mathrm{T}}$ 比较,在相同的条件下,高斯－塞德尔迭代比雅可比迭代得到的近似解的精度更高.

(2) 在 MATLAB 工作窗口输入程序

```
>> A=[3 4 -5 7;2 -8 3 -2;4 51 -13 16;7 -2 21 3];b=[5;2;-1;21];
X0=[0 0 0 0]';X=gsdddy(A,b,X0,inf,0.001,100)
```

运行后输出结果

```
请注意:因为对角矩阵 D 非奇异,所以此方程组有解.
请注意:高斯－塞德尔迭代的结果没有达到给定的精度,并且迭代次数已经超过
最大迭代次数 max1,方程组的精确解 jX 和迭代向量 X 如下:
jX =
    0.1821     -0.2571     0.7286     1.3036
X =   1.0e+142 *
```

```
0.2883    0.1062    0.3622    -3.1374
```

例 4.3.4 输电网络

一种大型输电网络可简化为图 4-1 所示电路,其中 $R_1, R_2, \cdots, R_n$ 表示负载电阻,$r_1, r_2, \cdots, r_n$ 表示线路内阻,设电源电压为 V,

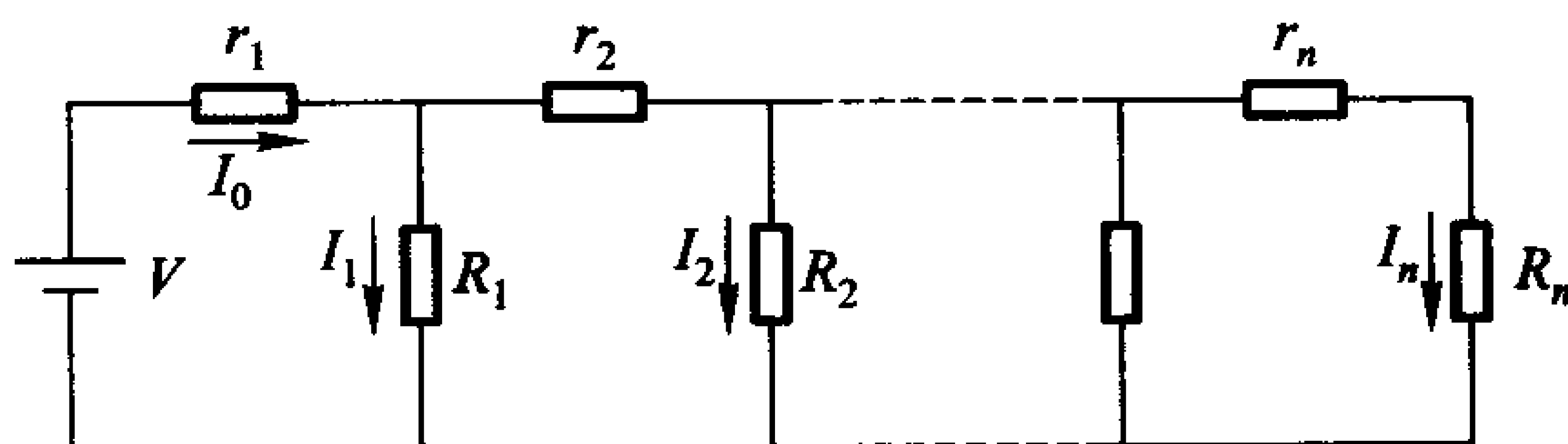


图 4-1 输电网络简化电路

(1) 列出求各负载上电流 $I_1, I_2, \cdots, I_n$ 的方程;

(2) 设 $R_1 = R_2 = \cdots = R_n = R$, $r_1 = r_2 = \cdots = r_n = r$,在 $r=1$, $R=6$, $V=18$, $n=10$ 的情况下求 $I_1, I_2, \cdots, I_n$ 及总电流 I_0;

(3) 讨论 $n \to \infty$ 的情况.

解 (1) 记 $r_1, \cdots, r_n$ 上的电流为 $i_1, \cdots, i_n$,根据电路中电流、电压的关系可以列出

$$\begin{cases} I_1 + i_2 = i_1, \\ I_2 + i_3 = i_2, \\ \cdots\cdots\cdots\cdots \\ I_{n-1} + i_n = i_{n-1}, \\ I_n = i_n \end{cases} \quad \text{和} \quad \begin{cases} r_1 i_1 + R_1 I_1 = V, \\ r_2 i_2 + R_2 I_2 = R_1 I_1, \\ \cdots\cdots\cdots\cdots \\ r_n i_n + R_n I_n = R_{n-1} I_{n-1}. \end{cases} \tag{4.18}$$

消去 $i_1, \cdots, i_n$ 得到

$$\begin{cases} (R_1 + r_1) I_1 + r_1 I_2 + \cdots + r_1 I_n = V, \\ -R_1 I_1 + (R_2 + r_2) I_2 + \cdots + r_2 I_n = 0, \\ \cdots\cdots\cdots\cdots \\ -R_{n-1} I_{n-1} + (R_n + r_n) I_n = 0. \end{cases} \tag{4.19}$$

记

$$\boldsymbol{R} = \begin{pmatrix} R_1 + r_1 & r_1 & r_1 & \cdots & r_1 & r_1 \\ -R_1 & R_2 + r_2 & r_2 & \cdots & r_2 & r_2 \\ & -R_2 & R_3 + r_3 & \cdots & r_3 & r_3 \\ & & \ddots & \ddots & \vdots & \vdots \\ & & & & -R_{n-1} & R_n + r_n \end{pmatrix},$$

$$\boldsymbol{I}=(I_1,I_2,\cdots,I_n)^{\mathrm{T}},\boldsymbol{E}=(V,0,\cdots,0)^{\mathrm{T}},$$

则方程(4.19)表为

$$\boldsymbol{RI}=\boldsymbol{E}. \tag{4.20}$$

(4.19)式或(4.20)式即为求解电流 $I_1,\cdots,I_n$ 的方程.

(2) 将已知条件输入,编写以下程序,并注意到总电流 I_0 为各负载上电流之和.

```
>> r=1;R=6;v=18;n=10; b1=sparse(1,1,v,n,1);
b=full(b1); a1=triu(r*ones(n,n));
a2=diag(R*ones(1,n));
a3=-tril(R*ones(n,n),-1)+tril(R*ones(n,n),-2);
a=a1+a2+a3; I=a\b; I0=sum(I)
```

计算结果如表 4－6 所示 (作为比较,表中还列出了 $n=20$ 的结果):

表 4－6 输电网络电流计算结果

k	0	1	2	3	4	5	6	7	8	9	10
$I_k(n=10)$	5.997	2.001	1.334	0.891	0.596	0.4000	0.270	0.186	0.132	0.101	0.087
$I_k(n=20)$	6.000	2.000	1.333	0.889	0.593	0.395	0.263	0.176	0.117	0.078	0.052
k		11	12	13	14	15	16	17	18	19	20
$I_k(n=20)$		0.035	0.023	0.015	0.010	0.007	0.005	0.003	0.002	0.002	0.002

可以看出,从 $n=10$ 到 $n=20$,总电流 I_0 几乎不变,$I_1\sim I_5$ 变化也很小,并且 I_{k+1} 差不多是 I_k 的 2/3 倍. 如果 n 增加到 50,100,可以得到类似的结论. 为证实这一结论需要研究 $n\to\infty$ 的情况.

(3) 从上面的计算看出,$n(n>10)$ 增大时总电流 I_0 几乎不变,说明总电阻增加很少. 为求出总电阻 R_0 考察图 4－2 所示的第 n 段电路(从右向左数)和第 $n+1$ 段电路的等效电阻 $R_0(n)$ 和 $R_0(n+1)$,有

$$R_0(n+1)=r+\frac{R\cdot R_0(n)}{R+R_0(n)},$$

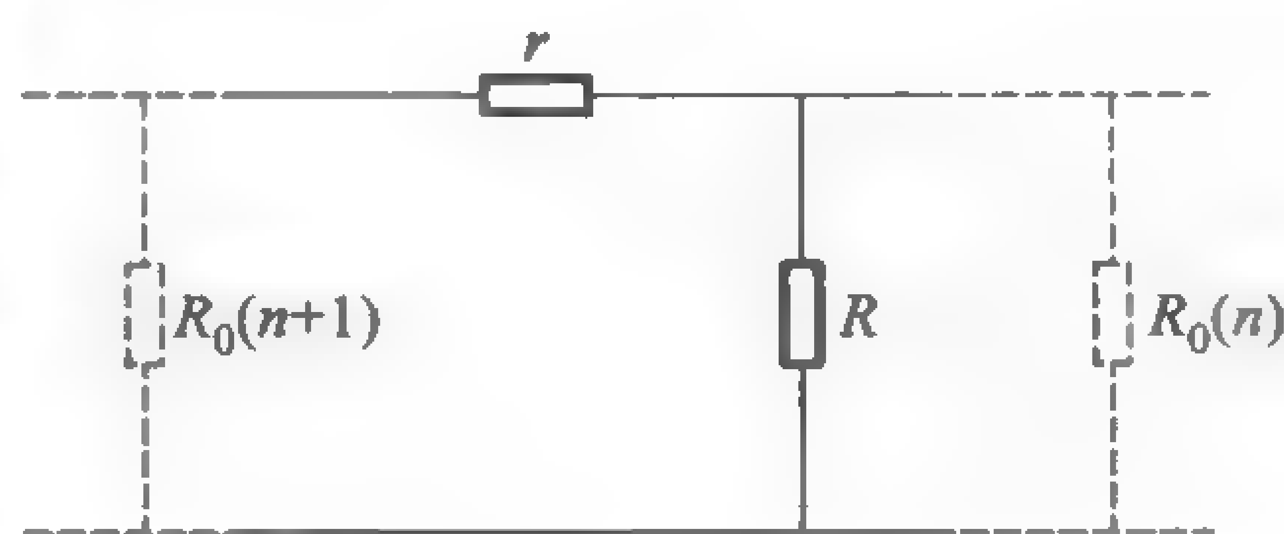


图 4－2 输电网络的等效电阻

以 $R_0(1)=R+r$ 代入可计算 $R_0(n)$,当 $n\to\infty$ 时记为 R_0,满足 $R_0=r+\dfrac{R\cdot R_0}{R+R_0}$,即 $R_0^2-rR_0-rR=0$,由此解得

$$R_0=\frac{r+\sqrt{r^2+4rR}}{2}=3\quad(r=1,R=6),$$

于是总电流 $I_0=i_1=\frac{V}{R_0}=\frac{18}{3}=6$. 而 $I_1=\frac{R_0}{R+R_0}\cdot i_1=\frac{1}{3}\cdot 6=2, i_2=\frac{R}{R+R_0}\cdot i_1$,

$I_2=\frac{R_0}{R+R_0}\cdot i_2=\frac{R}{R+R_0}\cdot I_1=\frac{2}{3}I_1,\cdots$,不难推出

$$I_{n+1}=\frac{R}{R+R_0}\cdot I_n=\frac{2}{3}I_n\quad(R=6,R_0=3)$$

证实了前面的计算结果.

(二) 高斯 - 塞德尔迭代公式的 MATLAB 程序

根据高斯 - 塞德尔迭代公式(4.16)、(4.17)、定理4.3 和公式(4.14),现提供一个名为 gsdd.m 的 M 文件如下

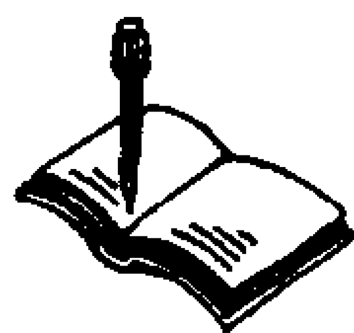
用高斯 - 塞德尔迭代解线性方程组 $AX=b$ 的 MATLAB 主程序

输入的量:线性方程组 $AX=b$ 的系数矩阵 A 和 b, 初始向量 X_0,范数的名称 $P=1,2,$inf, 或 'fro.',近似解 X 的误差(精度)*wucha* 和迭代的最大次数 *max*1.

输出的量:系数矩阵 $A=(a_{ij})_{n\times n}$的 $a=\sum_{\substack{j=1\\ i\neq k}}^{n}|a_{kj}|-|a_{kk}|\ (k=1,2,\cdots,n)$ 的值和有关高斯 - 塞德尔迭代收敛性的相关信息及其 $AX=b$ 的精确解 jX 和近似解 X.

```
function X = gsdd(A,b,X0,P,wucha,max1)
[n m] = size(A);
for j = 1:m
        a(j) = sum(abs(A(:,j))) -2 * (abs(A(j,j)));
end
for i = 1:n
  if a(i) > =0
            disp('请注意:系数矩阵 A 不是严格对角占优的,此高斯 - 塞德
尔迭代不一定收敛')
      return
  end
end
if a(i) <0
         disp('请注意:系数矩阵 A 是严格对角占优的,此方程组有唯一解,且
高斯 - 塞德尔迭代收敛 ')
end
```

```
for k = 1 :max1
        for j = 1:m
            if j = =1
                X(1) = (b(1) - A(1,2:m) * X0(2:m)) /A(1,1)
            end
            if j = =m
                X(m) = (b(m) - A(m,1:M1) * X(1:M1)') /A(m,m);
            end
            for j = 2:M1
X(j) = (b(j) - A(j,1:j-1) * X(1:j-1) - A(j,j+1:m) * X(j+1:m)) /A
(j,j);
            end
        end
          djwcX = norm(X' - X0,P);
          xdwcX = djwcX /(norm(X',P) + eps); X0 = X';X1 = A \b;
        if (djwcX < wucha) |(xdwcX < wucha)
              disp('请注意:高斯－塞德尔迭代收敛,此方程组的精确解 jX
和近似解 X 如下:')
                return
        end
 end
 if (djwcX > wucha)&(xdwcX > wucha)
       disp('请注意:高斯－塞德尔迭代次数已经超过最大迭代次数 max1 ')
 end
 a,X = X;jX = X1 ',
```



习 题 4.3

1. 编写雅可比迭代和高斯－塞德尔迭代的程序.

2. 用雅可比迭代和高斯－塞德尔迭代解方程组 $\boldsymbol{Ax}=\boldsymbol{b}$（$\boldsymbol{A}$ 如下,$\boldsymbol{b}$ 任意）,比较、分析其结果（包括迭代法收敛或不收敛的原因）.

(1) $\boldsymbol{A}=\begin{pmatrix} 9.362 & -2.701 & 0.384 \\ -5.980 & 7.248 & 0.682 \\ -1.370 & 3.086 & 8.241 \end{pmatrix}$;　　(2) $\boldsymbol{A}=\begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix}$;

(3) $\boldsymbol{A}=\begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix}$;　　(4) $\boldsymbol{A}$ 为 n 阶希尔伯特矩阵，$n=3\sim10$.

3. 用高斯-塞德尔迭代定义的 MATLAB 主程序解下列线性方程组.

(1) $\begin{cases} 11x_1 - x_2 - 2x_3 = 7.2, \\ -x_1 + 10x_2 - 2x_3 = 8.3, \\ -x_1 - x_2 + 0.5x_3 = 4.2; \end{cases}$ (2) $\begin{cases} 4x_1 + 4x_2 - 5x_3 + 7x_4 = 5, \\ 2x_1 - 8x_2 + 3x_3 - 2x_4 = 2, \\ 4x_1 + 51x_2 - 13x_3 + 16x_4 = -1, \\ 7x_1 - 2x_2 + 21x_3 + 3x_4 = 21. \end{cases}$

取初始值$(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0,0,0)$,要求当$\| \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \|_\infty < 10^{-5}$时,迭代终止.

4. 设有线性方程组

$$\begin{cases} 9x_1 + 2x_2 + 3x_3 = 1, \\ 2x_1 + 8x_2 + 2x_3 = 1, \\ 3x_1 + x_2 + 7x_3 = 2, \end{cases}$$

写出用雅可比迭代法和高斯-塞德尔迭代法解该方程组的迭代公式. 取初始值$(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0,0,0)$,求出方程组的近似值,要求当$\| \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \|_\infty < 10^{-5}$时,迭代终止.

4.4 解方程组的超松弛迭代法及其 MATLAB 程序

用雅可比迭代法和高斯-塞德尔迭代法解线性方程组时,有时收敛速度很慢,为了提高收敛速度,我们介绍超松弛迭代法,它是对高斯-塞德尔迭代的一种加速方法,适用于大型稀疏矩阵方程组的求解.

4.4.1 超松弛迭代法

用迭代法解方程组 $\boldsymbol{AX} = \boldsymbol{b}$ 时,我们将 $\boldsymbol{A}$ 分解为 $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}$, 其中 $\boldsymbol{D}, \boldsymbol{L}, \boldsymbol{U}$ 分别为对角矩阵,下三角形矩阵和上三角形矩阵. 我们给出的高斯-塞德尔迭代法是利用

$$\boldsymbol{DX}^{(k+1)} = \boldsymbol{LX}^{(k+1)} + \boldsymbol{UX}^{(k)} + \boldsymbol{b} \tag{4.21}$$

进行的. 如果将(4.21)式左端得到的 $\boldsymbol{X}^{(k+1)}$ 只作为中间结果,记作 $\widetilde{\boldsymbol{X}}^{(k+1)}$,而用它与 $\boldsymbol{X}^{(k)}$ 加权平均作为第 $k+1$ 步迭代的最终结果,即

$$\boldsymbol{X}^{(k+1)} = \omega \widetilde{\boldsymbol{X}}^{(k+1)} + (1 - \omega) \boldsymbol{X}^{(k)}, \tag{4.22}$$

ω 为**加权因子**,则有

$$\boldsymbol{DX}^{(k+1)} = \omega \boldsymbol{LX}^{(k+1)} + [\omega \boldsymbol{U} + (1 - \omega) \boldsymbol{D}] \boldsymbol{X}^{(k)} + \omega \boldsymbol{b},$$

即

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{B}_\omega \boldsymbol{X}^{(k)} + \boldsymbol{f}_\omega \quad (k = 0, 1, 2, \cdots), \tag{4.23}$$

其中 $\boldsymbol{B}_\omega = (\boldsymbol{D} - \omega \boldsymbol{L})^{-1} [\omega \boldsymbol{U} + (1 - \omega) \boldsymbol{D}], \boldsymbol{f}_\omega = \omega (\boldsymbol{D} - \omega \boldsymbol{L})^{-1} \boldsymbol{b},$

$\omega > 1$ 时(4.23)式称为**超松弛迭代**,$\omega < 1$ 时称为**低松弛迭代**,$\omega = 1$ 时则退化为**高斯-塞德尔迭代**.

4.4.2 超松弛迭代法收敛性及其 MATLAB 程序

定理 4.5 迭代公式(4.23)收敛的充要条件是谱半径 $\rho(\boldsymbol{B}_\omega)<1$.

但是对于给定的 $\boldsymbol{A}$,ω 取值在什么范围内(4.23)式收敛以及 ω 取何值时收敛最快,是十分复杂的问题. 一个有实用价值的结果为.

定理 4.6 如果解方程组 $\boldsymbol{AX}=\boldsymbol{b}$ ($a_{ii}\neq 0, i=1,2,\cdots,n$)的迭代公式(4.23)收敛,则 $0<\omega<2$.

这个定理给出了超松弛迭代法收敛的必要条件. 这就是说只有松弛因子 ω 在(0,2)内取值时,超松弛迭代法才有可能收敛. 但是系数矩阵 $\boldsymbol{A}$ 是正定对称阵,且 $0<\omega<2$,则此迭代公式(4.23)一定收敛.

定理 4.7 如果线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的系数矩阵 $\boldsymbol{A}$ 是正定对称阵,且 $0<\omega<2$,则此方程组的迭代公式(4.23)收敛.

使用超松弛迭代法的关键是如何选择松弛因子 ω 的值,使得收敛的速度最快,这就是所谓的**最佳松弛因子问题**. 但是目前只有少数特殊类型的矩阵才能求出其最佳松弛因子. 一般采用逐步搜索的方法,即在(0,2)内选取一个或多个不同松弛因子 ω 的值进行试算,然后根据迭代过程收敛速度的快慢,逐步寻找最佳的 ω,最后固定下来继续迭代. 让我们从下例观察 ω 的取值对于收敛速度的影响.

根据定理 4.5 和谱半径定义,现提供一个名为 ddpbj.m 的 M 文件,用于判别超松弛迭代公式(4.23)产生的迭代序列的敛散性.

用谱半径判别超松弛迭代法产生的迭代序列的敛散性的 MATLAB 主程序

输入的量:线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的系数矩阵 $\boldsymbol{A}$ 和松弛因子 *om*;

输出的量:矩阵 $\boldsymbol{B}_\omega=(\boldsymbol{D}-\omega\boldsymbol{L})^{-1}[\omega\boldsymbol{U}+(1-\omega)\boldsymbol{D}]$的所有特征值和谱半径 $mH=\rho(\boldsymbol{B}_\omega)$及其有关超松弛迭代法产生的迭代序列的敛散性的相关信息.

```
function H = ddpbj(A,om)
      D = diag(diag(A));U = -triu(A,1);
      L = -tril(A,-1); iD = inv(D - om * L);
      B2 = iD * (om * U + (1 - om) * D);
      H = eig(B2);mH = norm(H,inf);
if mH > =1
          disp('请注意:因为谱半径不小于 1,所以超松弛迭代序列发散,谱半径
mH 和 B 的所有的特征值 H 如下:')
else
```

```
        disp('请注意:因为谱半径小于1,所以超松弛迭代序列收敛,谱半径mH
和B的所有的特征值H如下:')
    end
    mH
```

例 4.4.1 当取 $\omega=1.15,5$ 时,判别用超松弛迭代法解下列方程组产生的迭代序列是否收敛?

$$\begin{cases}5x_1+x_2-x_3-2x_4=4,\\2x_1+8x_2+x_3+3x_4=1,\\x_1-2x_2-4x_3-x_4=6,\\-x_1+3x_2+2x_3+7x_4=-3.\end{cases}$$

解 (1) 当取 $\omega=1.15$ 时,首先保存名为 ddpbj.m 的 M 文件,然后在 MATLAB 工作窗口输入程序

```
>> A=[5 1 -1 -2;2 8 1 3;1 -2 -4 -1;-1 3 2 7]; H=ddpbj(A,1.15)
```

运行后输出结果

```
    请注意:因为谱半径小于1,所以超松弛迭代序列收敛,谱半径mH和B的所有的
特征值H如下:
    mH =
        0.1596
    H =
    0.1049 + 0.1203i 0.1049 - 0.1203i -0.1295 + 0.0556i -0.1295
     -0.0556i
```

(2) 当取 $\omega=5$ 时,然后在 MATLAB 工作窗口输入程序

```
>> H=ddpbj(A, 5)
```

运行后输出结果

```
    请注意:因为谱半径不小于1,所以超松弛迭代序列发散,谱半径mH和B的所有
的特征值H如下:
    mH =
          14.1082
    H =
      -14.1082      -2.5107      0.5996 +2.6206i      0.5996 -2.6206i
```

例 4.4.2 在(0,2)内取不同的 ω 值,用超松弛迭代法解 $\boldsymbol{Ax}=\boldsymbol{b}$,并比较迭代过程收敛的速度的快慢,其中

$$\begin{cases}4x_1-x_2-x_3-x_4=1,\\-x_1+4x_2-x_3-x_4=1,\\-x_1-x_2+4x_3-x_4=1,\\-x_1-x_2-x_3+4x_4=1.\end{cases}$$

解 取 $\omega=0.75,1.0,1.25,1.5$ 和 $\boldsymbol{x}^{(0)}=(0,0,0,0)^{\mathrm{T}}$,用(4.23)式得超松弛迭代公式为

$$
\begin{cases}
x_1^{(k+1)}=x_1^{(k)}+\dfrac{\omega}{4}(-4x_1^{(k)}+x_2^{(k)}+x_3^{(k)}+x_4^{(k)}+1),\\
x_2^{(k+1)}=x_2^{(k)}+\dfrac{\omega}{4}(x_1^{(k+1)}-4x_2^{(k)}+x_3^{(k)}+x_4^{(k)}+1),\\
x_3^{(k+1)}=x_3^{(k)}+\dfrac{\omega}{4}(x_1^{(k+1)}+x_2^{(k+1)}-4x_3^{(k)}+x_4^{(k)}+1),\\
x_4^{(k+1)}=x_4^{(k)}+\dfrac{\omega}{4}(x_1^{(k+1)}+x_2^{(k+1)}+x_3^{(k+1)}-4x_4^{(k)}+1).
\end{cases}
$$

求解结果列入表 4-7(只给出 $\boldsymbol{x}^{(5)}$ 和 $\boldsymbol{x}^{(10)}$).

表 4-7

ω	$\boldsymbol{x}^{(5)}$	$\boldsymbol{x}^{(10)}$
0.75	(0.754 9, 0.773 2, 0.790 2, 0.806 0)	(0.948 4, 0.952 2, 0.955 8, 0.9591)
1.0	(0.916 5, 0.927 4, 0.937 0, 0.945 2)	(0.995 0, 0.995 6, 0.996 2, 0.996 7)
1.25	(0.998 3, 0.999 3, 0.998 9, 0.999 8)	(1.000 0, 1.000 0, 1.000 0, 1.000 0)
1.5	(0.975 9, 0.979 9, 0.974 9, 0.991 0)	(0.999 8, 0.999 2, 1.000 0, 0.999 9)

与精确解 $\boldsymbol{X}=(1,1,1,1)^{\mathrm{T}}$ 相比，可知 $\omega=1.25$ 时收敛最快.

4.4.3 超松弛迭代法的 MATLAB 程序

根据超松弛迭代公式(4.23)和定理 4.5,现提供一个名为 cscdd.m 的 M 文件如下:

> **用超松弛迭代法解线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的 MATLAB 主程序**
>
> 输入的量:线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的系数矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{b}$，初始向量 $\boldsymbol{X}$,范数的名称 $P=$ 1,2,inf,或 'fro.',松弛因子 *om*,近似解 $\boldsymbol{X}$ 的误差(精度)*wucha* 和迭代的最大次数 *max*1.
>
> 输出的量:谱半径 *mH*,以系数矩阵 $\boldsymbol{A}$ 的对角元构成的对角矩阵 $\boldsymbol{D}$、$\boldsymbol{A}$ 的上三角形矩阵 $\boldsymbol{U}$,但对角元为 0、$\boldsymbol{A}$ 的下三角形矩阵 $\boldsymbol{L}$,但对角元为 0，迭代次数 i,有关超松弛迭代收敛性的相关信息及其 $\boldsymbol{AX}=\boldsymbol{b}$ 的精确解 $\boldsymbol{jX}$ 和近似解 $\boldsymbol{X}$.

```
function X = cscdd (A,b,X,om,wucha,max1)
    D = diag(diag(A));U = -triu(A,1);
    L = -tril(A,-1); jX = A\b;[n m] = size(A);
    iD = inv(D - om * L); B2 = iD * (om * U + (1 - om) * D);
```

```
      H = eig(B2);mH = norm(H,inf);
    for k = 1:max1
      iD = inv(D - om * L); B2 = iD * (om * U + (1 - om) * D);
      f2 = om * iD * b; X1 = B2 * X + f2;
      X = X1; djwcX = norm(X1 - jX,inf); xdwcX = djwcX /(norm(X,inf) +
      eps);
      if (djwcX < wucha) |(xdwcX < wucha)
            disp('谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代
次数 i 如下:')
            mH,D,U,L,jX = jX', i = k - 1,
          return
         if i > max1
           disp('迭代次数已经超过最大迭代次数 max1,谱半径 mH,方程组的精
确解 jX,迭代次数 i 如下:')
           mH,D,U,L,jX = jX', i = k - 1,
     end
    end
   end
   if mH > =1
       disp('请注意:因为谱半径不小于 1,所以超松弛迭代序列发散.')
         disp('谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数
i 和迭代序列 X 如下:')
    i = k - 1,mH,D,U,L,jX,
    else
        disp('因为谱半径小于 1,所以超松弛迭代序列收敛,近似解 X 如下:')
    end
```

或

```
    function X = cscdd1 (A,b,X,om,wucha,max1)
      D = diag(diag(A));U = - triu(A,1);L = - tril(A, - 1); jX = A \b;
      [n m] = size(A);
      iD = inv(D - om * L); B2 = iD * (om * U + (1 - om) * D);
      H = eig(B2);mH = norm(H,inf);
    for k = 1:max1
       iD = inv(D - om * L); B2 = iD * (om * U + (1 - om) * D);
       f2 = om * iD * b; X1 = B2 * X + f2; X = X1; djwcX = norm(X1 - jX,
       inf);
       xdwcX = djwcX /(norm(X,inf) + eps);
    end
```

```
if mH > =1
    disp('请注意:因为谱半径不小于1,所以超松弛迭代序列发散.谱半径 mH,
    A 的分解矩阵 D,U,L 和方程组的精确解 jX 和近似解 X 如下:')
else
     disp('请注意:因为谱半径小于1,所以超松弛迭代序列收敛.')
     if (djwcX < wucha) |(xdwcX < wucha)
            disp('谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX 和
近似解 X 如下: ')
         mH,D,U,L,jX = jX',
     else
       disp('迭代次数已经超过最大迭代次数 max1,谱半径 mH,方程组的精
确解 jX 和迭代向量 X 如下: ')
         mH,D,U,L,X = X1 ';jX = jX'
         return
     end
  end
```

例 4.4.3 用超松弛迭代法(取 $\omega = 1.15$ 和 5)解例 4.4.1 中的线性方程组.

解 (1) 当取 $\omega = 1.15$ 时,首先保存名为 cscdd.m 的 M 文件,然后在 MATLAB 工作窗口输入程序

```
>> A = [5 1 -1 -2;2 8 1 3;1 -2 -4 -1;-1 3 2 7];b = [4;1;6;-3];
   X = [0 0 0 0]';X = cscdd (A,b,X,1.15,0.001,100),
```

运行后输出结果

```
谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数 i 如下:
mH -
    0.1596
D =
   5  0    0  0
   0  8    0  0
   0  0   -4  0
   0  0    0  7
U =
   0   -1    1    2
   0    0   -1   -3
   0    0    0    1
   0    0    0    0
L =
```

```
     0     0     0   0
    -2     0     0   0
    -1     2     0   0
     1    -3    -2   0
jX =
   0.4491    0.2096    -1.4850    -0.0299
i =
     3
```

因为谱半径小于1,所以超松弛迭代序列收敛,近似解 X 如下:

```
X =
     0.4484
     0.2100
    -1.4858
    -0.0303
```

(2) 当取 $\omega=5$ 时,首先保存名为 cscdd.m 的 M 文件,然后在 MATLAB 工作窗口输入程序

```
>>A=[5 1 -1 -2;2 8 1 3;1 -2 -4 -1;-1 3 2 7];b=[4;1;6;-3];
  X=[0 0 0 0]';X=cscdd(A,b,X,5,0.001,100),
```

运行后输出结果如下:

请注意:因为谱半径不小于1,所以超松弛迭代序列发散.

谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数 i 和迭代序列 X 如下:

```
i =
     99
mH =
   14.1082
D =
   5   0    0   0
   0   8    0   0
   0   0   -4   0
   0   0    0   7
U =
   0   -1    1    2
   0    0   -1   -3
   0    0    0    1
   0    0    0    0
L =
```

```
     0     0     0     0
    -2     0     0     0
    -1     2     0     0
     1    -3    -2     0
jX =
      0.4491
      0.2096
     -1.4850
     -0.0299
X -1.0e+114 *
   -0.3122
   1.0497
   -3.7174
   3.9615
```

例 4.4.4 在(0,2)内取 $\omega=0.15,0.45,0.75,1.0,1.25,1.5,1.75$ 和$x^{(0)}=(0,0,0,0)^T$,用超松弛迭代法解 $Ax=b$,误差为 0.001,并比较迭代过程收敛的速度的快慢,其中

$$\begin{cases}5x_1-x_2-x_3-2x_4=2,\\-x_1+5x_2-x_3-x_4=1,\\-x_1-x_2+5x_3-x_4=1,\\-x_1-x_2+2x_3+7x_4=1.\end{cases}$$

解 下面分别用程序 cscdd.m 和程序 cscdd1.m 计算.

(1) 用程序 cscdd.m 计算.

在(0,2)内取 $\omega=0.15,0.45,0.75,1.0,1.25,1.5,1.75$ 值和 $x^{(0)}=(0,0,0,0)^T$,保存名为 cscdd.m 的 M 文件,然后在 MATLAB 工作窗口输入程序

```
>> A=[5 -1 -1 -2;-1 5 -1 -1;-1 -1 5 -1;-1 -1 2 7];
b=[2;1;1;1]; X=[0 0 0 0]';
X1=cscdd(A,b,X, 0.15,0.001,500),
X2=cscdd(A,b,X,0.45,0.001,500),
X3=cscdd(A,b,X,0.75,0.001,500),
X4=cscdd(A,b,X,1.0,0.001,500),
X5=cscdd(A,b,X,1.25,0.001,500), X6=cscdd(A,b,X,1.5,0.001,
500),
X7=cscdd(A,b,X,1.75,0.001,500),
```

运行后输出结果

```
谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数 i 如下:
mH=0.9086
```

```
L =0  0   0   0    D =5  0  0  0    u =0  1  1  2
   1  0   0   0        0  5  0  0        0  0  1  1
   1  1   0   0        0  0  5  0        0  0  0  1
   1  1  -2   0        0  0  0  7        0  0  0  0
jX =   0.6504    0.4553    0.4553    0.1707
i =   67
```

因为谱半径小于 1,所以超松弛迭代序列收敛,近似解 X 如下:

```
   X1 =  0.6494    0.4543    0.4544    0.1707
```

谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数 i 如下:

```
mH =  0.7138,i = 19
```

因为谱半径小于 1,所以超松弛迭代序列收敛,近似解 X 如下:

```
X2 =  0.6496    0.4546    0.4547    0.1707
```

谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数 i 如下:

```
mH =  0.4908,i = 9
```

因为谱半径小于 1,所以超松弛迭代序列收敛,近似解 X 如下:

```
X3 =  0.6498    0.4548    0.4549    0.1707
```

谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数 i 如下:

```
mH =  0.2516,i = 5
```

因为谱半径小于 1,所以超松弛迭代序列收敛,近似解 X 如下:

```
X4 =  0.6501    0.4551    0.4552    0.1707
```

谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数 i 如下:

```
mH =  0.4541,i = 4
```

因为谱半径小于 1,所以超松弛迭代序列收敛,近似解 X 如下:

```
X5 =  0.6508    0.4552    0.4557    0.1706
```

谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数 i 如下:

```
mH =  0.8580,i = 26
```

因为谱半径小于 1,所以超松弛迭代序列收敛,近似解 X 如下:

```
X6 =  0.6508    0.4554    0.4563    0.1700
```

请注意:因为谱半径不小于 1,所以超松弛迭代序列发散.

谱半径 mH,A 的分解矩阵 D,U,L 和方程组的精确解 jX,迭代次数 i 和迭代序列 X 如下:

```
i =499,mH =1.2976
jX =  0.6504    0.4553    0.4553    0.1707
X7 =1.0e +055 *
     -1.1930    -0.6894    -3.4948    3.0254
```

以上的运行结果与方程组的精确解 $jX = (0.6504, 0.4553, 0.4553, 0.1707)^T$ 的误差为 0.001 的近似解 $x^{(10)}$ 和迭代次数如表 4－8 所示.

表 4-8 与方程组的精确解 jX 的误差为 0.001 的近似解 $x^{(10)}$ 和迭代次数

ω	迭代次数 i	谱半径 mH	解的迭代序列 $x^{(10)}$,
0.15	67	0.908 6	(0.649 4, 0.454 3, 0.454 4, 0.170 7)
0.45	19	0.713 8	(0.649 6, 0.454 6, 0.454 7, 0.170 7)
0.75	9	0.490 8	(0.649 8, 0.454 8, 0.454 9, 0.170 7)
1.0	5	0.251 6	(0.650 1, 0.455 1, 0.455 2, 0.170 7)
1.25	4	0.454 1	(0.650 8, 0.455 2, 0.455 7, 0.170 6)
1.5	26	0.858 0	(0.650 8, 0.455 4, 0.456 3, 0.170 0)
1.75	449	1.297 6	$(-1.193\ 0\times10^{55},\ -0.689\ 4\times10^{55},\ -3.494\ 8\times10^{55},\ 3.025\ 4\times10^{55})$

由表 4-8 可见，$\omega=1.25$ 时，收敛速度最快，迭代 4 次，近似解 $X=(0.650\ 8, 0.455\ 2, 0.455\ 7, 0.170\ 6)^T$ 与方程组的精确解 $jX=(0.650\ 4, 0.455\ 3, 0.455\ 3, 0.170\ 7)^T$ 的误差为 0.001. $\omega=1.0$ 时，收敛速度也快，虽然迭代 5 次，但是近似解 $X=(0.650\ 1, 0.455\ 1, 0.455\ 2, 0.170\ 7)^T$ 与方程组的精确解 $jX=(0.650\ 4, 0.455\ 3, 0.455\ 3, 0.170\ 7)^T$ 的误差为比 $\omega=1.25$ 时小. 只有当 $\omega=1.75$ 时，迭代序列发散. 迭代序列 $X=(-1.193\ 0\times10^{55}, -0.689\ 4\times10^{55}, -3.494\ 8\times10^{55}, 3.025\ 4\times10^{55})^T$ 与方程组的精确解 $jX=(0.650\ 4, 0.455\ 3, 0.455\ 3, 0.170\ 7)^T$ 差得太远.

(2) 用程序 cscdd1.m 计算. 最大迭代次数 $max1=5$ 时，在 (0,2) 内取 $\omega=0.15, 0.45, 0.75, 1.0, 1.25, 1.5, 1.75$ 值和 $x^{(0)}=(0,0,0,0)^T$，保存名为 cscdd1.m 的M 文件，然后在 MATLAB 工作窗口输入程序

```
>> A=[5 -1 -1 -2; -1 5 -1 -1; -1 -1 5 -1;-1 -1 2 7];b=[2;
1;1;1];
X=[0 0 0 0]'; X1=cscdd1(A,b,X, 0.15,0.001,5),
X2=cscdd1(A,b,X,0.45,0.001,5), X3=cscdd1(A,b,X,0.75,0.001,5),
X4=cscdd1(A,b,X,1.0,0.001,5), X5=cscdd1(A,b,X,1.25,0.001,5),
X6=cscdd1(A,b,X,1.5,0.001,5), X7=cscdd1(A,b,X,1.75,0.001,5),
```

运行后输出结果

```
请注意:因为谱半径小于1,所以超松弛迭代序列收敛.
迭代次数已经超过最大迭代次数 max1,谱半径 mH,方程组的精确解 jX 和迭代向
量 X 如下:
mH = 0.9086
........................
jX =  0.6504    0.4553    0.4553    0.1707
```

X1 = 0.2482 0.1435 0.1465 0.0851

请注意:因为谱半径小于 1,所以超松弛迭代序列收敛.

迭代次数已经超过最大迭代次数 max1,**谱半径** mH,**方程组的精确解** jX **和迭代向量** X **如下:**

mH = 0.7138

X2 = 0.5247 0.3513 0.3594 0.1545

请注意:因为谱半径小于 1,所以超松弛迭代序列收敛.

迭代次数已经超过最大迭代次数 max1,**谱半径** mH,**方程组的精确解** jX **和迭代向量** X **如下:**

mH = 0.4908

X3 = 0.6287 0.4388 0.4420 0.1682

请注意:因为谱半径小于 1,所以超松弛迭代序列收敛.

迭代次数已经超过最大迭代次数 max1,**谱半径** mH,**方程组的精确解** jX **和迭代向量** X **如下:**

mH = 0.2516

X4 = 0.6493 0.4546 0.4548 0.1706

请注意:因为谱半径小于 1,所以超松弛迭代序列收敛.

谱半径 mH,A **的分解矩阵** D,U,L **和方程组的精确解** jX **和近似解** X **如下:**

mH = 0.4541

X5 = 0.6508 0.4552 0.4557 0.1706

请注意:因为谱半径小于 1,所以超松弛迭代序列收敛.

迭代次数已经超过最大迭代次数 max1,**谱半径** mH,**方程组的精确解** jX **和迭代向量** X **如下:**

mH - 0.8580

X6 = 0.6713 0.4598 0.4847 0.1501

请注意:因为谱半径不小于 1,所以超松弛迭代序列发散.谱半径 mH,A **的分解矩阵** D,U,L **和方程组的精确解** jX **和近似解** X **如下:**

X7 = 0.8188 0.5207 0.7873 -0.1228

为了比较迭代过程收敛的速度的快慢,在 MATLAB 工作窗口输入程序

```
>> jX = [0.6504   0.4553   0.4553   0.1707];
X1 = [0.2482   0.1435   0.1465   0.0851];
X2 = [0.5247   0.3513   0.3594   0.1545];
X3 = [0.6287   0.4388   0.4420   0.1682];
X4 = [0.6493   0.4546   0.4548   0.1706];
X5 = [0.6508   0.4552   0.4557   0.1706];
X6 = [0.6713   0.4598   0.4847   0.1501];
X7 = [0.8188   0.5207   0.7873   -0.1228];
nX1 = norm(X1 - jX),nX2 = norm(X2 - jX),
```

```
nX3 = norm(X3 - jX), nX4 = norm(X4 - jX),
nX5 = norm(X5 - jX),nX6 = norm(X6 - jX),
nX7 = norm(X7 - jX),Y = [nX1,nX2,
nX3,nX4,nX5,nX6]; maxXi = max(Y), minXi = min(Y)
```

运行后输出结果

```
nX1 = 0.6014,nX2 = 0.1899,nX3 = 0.0304,
nX4 = 0.0014, nX5 = 5.8310e - 004,
nX6 = 0.0418,nX7 = 0.4785, maxXi = 0.6014,minXi = 5.8310e - 004
```

显然,当 $\omega=1.25$ 时,收敛速度最快;$\omega=0.15$ 时,收敛速度最慢;当 $\omega=1.75$ 时,因为谱半径不小于 1,所以超松弛迭代序列发散.

例 4.4.5 分别用雅可比迭代法,高斯 - 塞德尔迭代法和超松弛迭代法(取 $\omega=1.15$)解下面的线性方程组,当 $\max\limits_{1\leqslant i\leqslant 4}|x_i^{(k+1)}-x_i^{(k)}|<10^{-5}$时迭代停止.

$$\begin{cases} 5x_1+x_2-x_3-2x_4=-2,\\ 2x_1+8x_2+x_3+3x_4=-6,\\ x_1-2x_2-4x_3-x_4=6,\\ -x_1+3x_2+2x_3+7x_4=12. \end{cases}$$

解 取 $X^{(0)}=(0\ 0\ 0\ 0)^{\mathrm{T}}$,超松弛迭代公式为

$$\begin{cases} x_1^{(k+1)}=x_1^{(k)}+\dfrac{\omega}{5}(-5x_1^{(k)}-x_2^{(k)}+x_3^{(k)}+2x_4^{(k)}-2),\\ x_2^{(k+1)}=x_2^{(k)}+\dfrac{\omega}{8}(-2x_1^{(k+1)}-8x_2^{(k)}-x_3^{(k)}-3x_4^{(k)}-6),\\ x_3^{(k+1)}=x_3^{(k)}-\dfrac{\omega}{4}(-x_1^{(k+1)}+2x_2^{(k+1)}+4x_3^{(k)}+x_4^{(k)}+6),\\ x_4^{(k+1)}=x_4^{(k)}+\dfrac{\omega}{7}(x_1^{(k+1)}-3x_2^{(k+1)}-2x_3^{(k+1)}-7x_4^{(k)}+12). \end{cases}$$

迭代 8 次,得方程组的近似解为

$$X^{(8)}=(0.999\,996\,5,\ -1.999\,997\,0,\ -1.000\,001\,0, 2.999\,999\,0)^{\mathrm{T}}.$$

高斯 - 塞德尔迭代公式为

$$\begin{cases} x_1^{(k+1)}=x_1^{(k)}+\dfrac{1}{5}(-5x_1^{(k)}-x_2^{(k)}+x_3^{(k)}+2x_4^{(k)}-2),\\ x_2^{(k+1)}=x_2^{(k)}+\dfrac{1}{8}(-2x_1^{(k+1)}-8x_2^{(k)}-x_3^{(k)}-3x_4^{(k)}-6),\\ x_3^{(k+1)}=x_3^{(k)}-\dfrac{1}{4}(-x_1^{(k+1)}+2x_2^{(k+1)}+4x_3^{(k)}+x_4^{(k)}+6),\\ x_4^{(k+1)}=x_4^{(k)}+\dfrac{1}{7}(x_1^{(k+1)}-3x_2^{(k+1)}-2x_3^{(k+1)}-7x_4^{(k)}+12). \end{cases}$$

迭代 14 次,得方程组的近似解为

$$X^{(14)} = (0.999\,996\,6,\ -1.999\,997\,0,\ -1.000\,001\,0, 2.999\,999\,0)^{T}$$

雅可比迭代公式为

$$\begin{cases} x_1^{(k+1)} = x_1^{(k)} + \dfrac{1}{5}(-5x_1^{(k)} - x_2^{(k)} + x_3^{(k)} + 2x_4^{(k)} - 2), \\ x_2^{(k+1)} = x_2^{(k)} + \dfrac{1}{8}(-2x_1^{(k)} - 8x_2^{(k)} - x_3^{(k)} - 3x_4^{(k)} - 6), \\ x_3^{(k+1)} = x_3^{(k)} - \dfrac{1}{4}(-x_1^{(k)} + 2x_2^{(k)} + 4x_3^{(k)} + x_4^{(k)} + 6), \\ x_4^{(k+1)} = x_4^{(k)} + \dfrac{1}{7}(x_1^{(k)} - 3x_2^{(k)} - 2x_3^{(k)} - 7x_4^{(k)} + 12). \end{cases}$$

迭代 24 次,得方程组的近似解为

$$X^{(24)} = (0.999\,994\,1,\ -1.999\,995\,0,\ -1.000\,004\,0, 2.999\,999\,0)^{T}$$



习 题 4.4

1. 取 $\omega = 1.15, 5$ 时,判别用超松弛迭代法解下列方程组产生的迭代序列是否收敛?

$$\begin{cases} 7x_1 + x_2 - x_3 - 2x_4 = 4, \\ 2x_1 + 8x_2 + x_3 + 3x_4 = 1, \\ x_1 - 2x_2 - 4x_3 - x_4 = 6, \\ -x_1 + 3x_2 + 2x_3 + 7x_4 = -3. \end{cases}$$

2. 在(0,2)内取 $\omega = 0.75, 1.0, 1.25, 1.5$ 和 $x^{(0)} = (0,0,0,0)^{T}$,用超松弛迭代法解 $Ax = b$,误差为 0.001. 并比较迭代过程收敛的速度的快慢,其中

$$(1)\begin{cases} 5x_1 - 3x_2 - 7x_3 - 2x_4 = 2, \\ -2x_1 + 5x_2 - x_3 - x_4 = 21, \\ -9x_1 - x_2 + 5x_3 - x_4 = 1, \\ -3x_1 - x_2 + 2x_3 + 7x_4 = 2.5; \end{cases} \qquad (2)\begin{cases} 7x_1 - x_2 - x_3 - 2x_4 = 2, \\ -x_1 + 5x_2 - x_3 - x_4 = 1, \\ -x_1 - x_2 + 5x_3 - x_4 = 3, \\ -x_1 - x_2 + 2x_3 + 7x_4 = 1. \end{cases}$$

3. 分别用雅可比迭代法,高斯-塞德尔迭代法和超松弛迭代法(取 $\omega = 1.15$)解下面的线性方程组,当 $\max\limits_{1 \le i \le 4} |x_i^{(k+1)} - x_i^{(k)}| < 10^{-5}$ 时迭代停止.

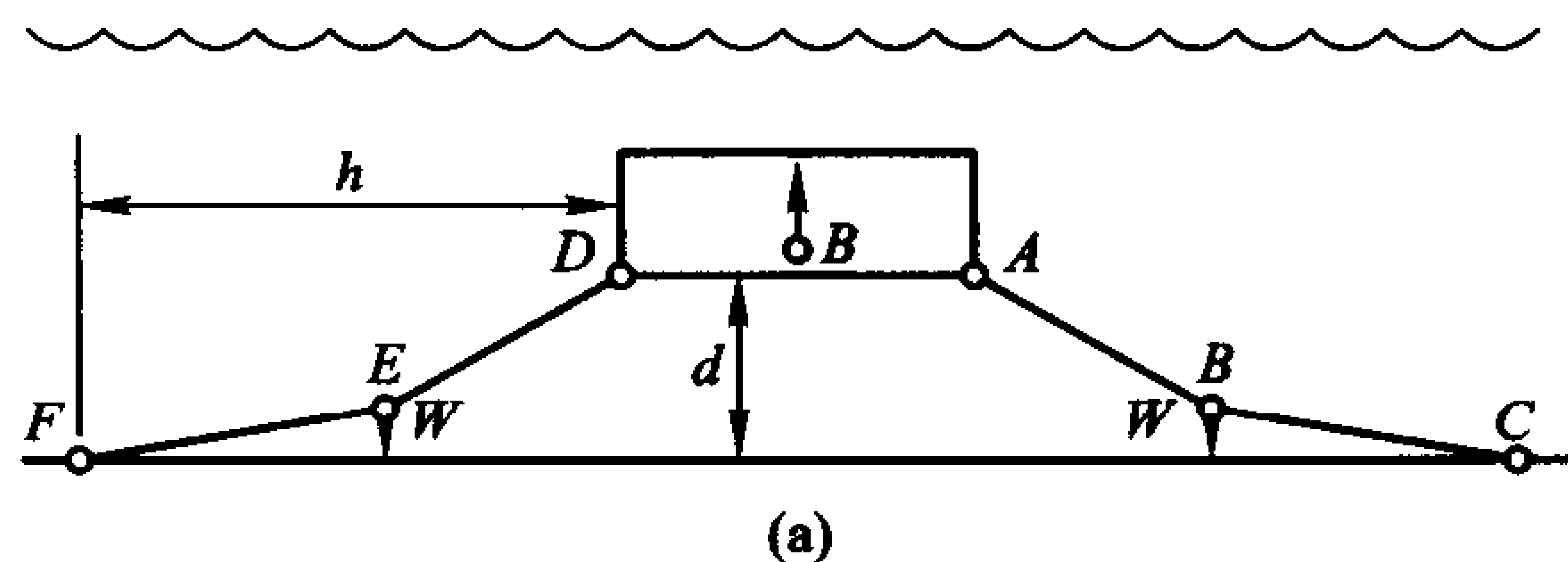
$$\begin{cases} 5x_1 + x_2 - x_3 - 2x_4 = -2, \\ 2x_1 + 8x_2 + x_3 + 3x_4 = -6, \\ x_1 - 2x_2 - 4x_3 - x_4 = 6, \\ -x_1 + 3x_2 + 2x_3 + 7x_4 = 12. \end{cases}$$

4. 水下一浮体 M 被两条缆绳 ABC 和 DEF 固定,浮体受上浮力 B,在 B, E 处各挂一重物 W, $AB(DE)$ 及 $BC(EF)$ 与水平线夹角分别为 20°和 10°(图 4-3). 设 $B = 10$ kN,

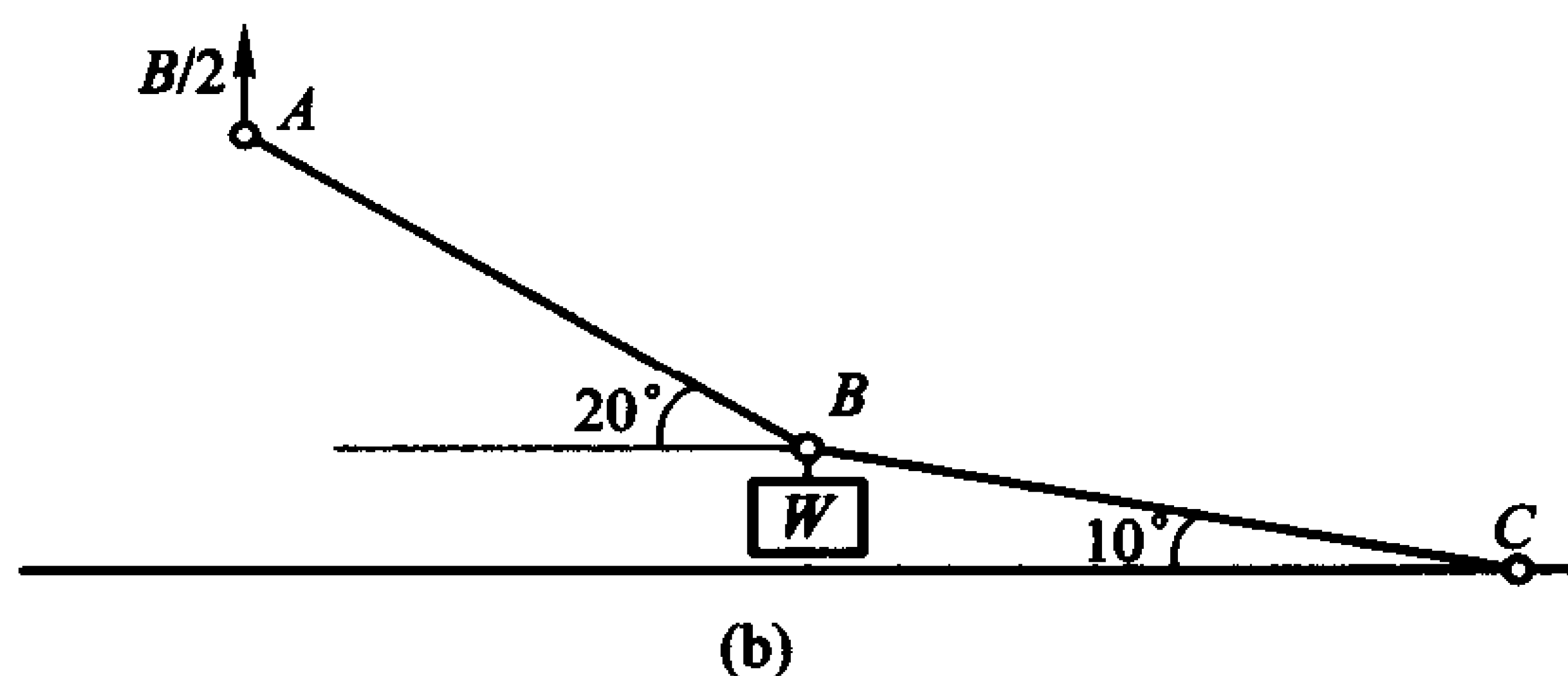
(1) 求 AB (DE)和 BC(EF)段所受张力 T_1,T_2 及 W;

(2) 若要求 $h=8$, $d=2$, 求 AB (DE)和 BC(EF)段长度;

(3) 如图 4－4 所示,若缆绳分为 AB, BC, CD 三段,与水平线夹角分别为 40°,30°和 20°,求这三段所受张力及 W_1, W_2.

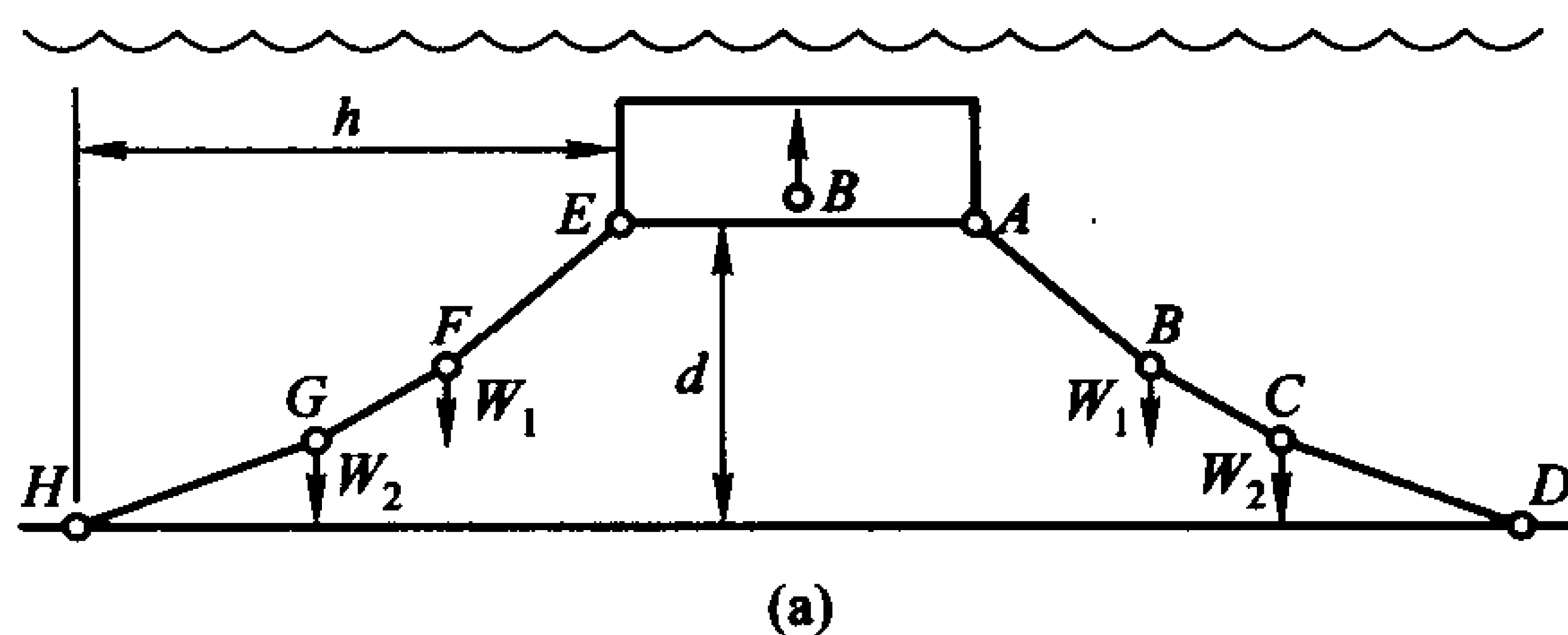


(a)

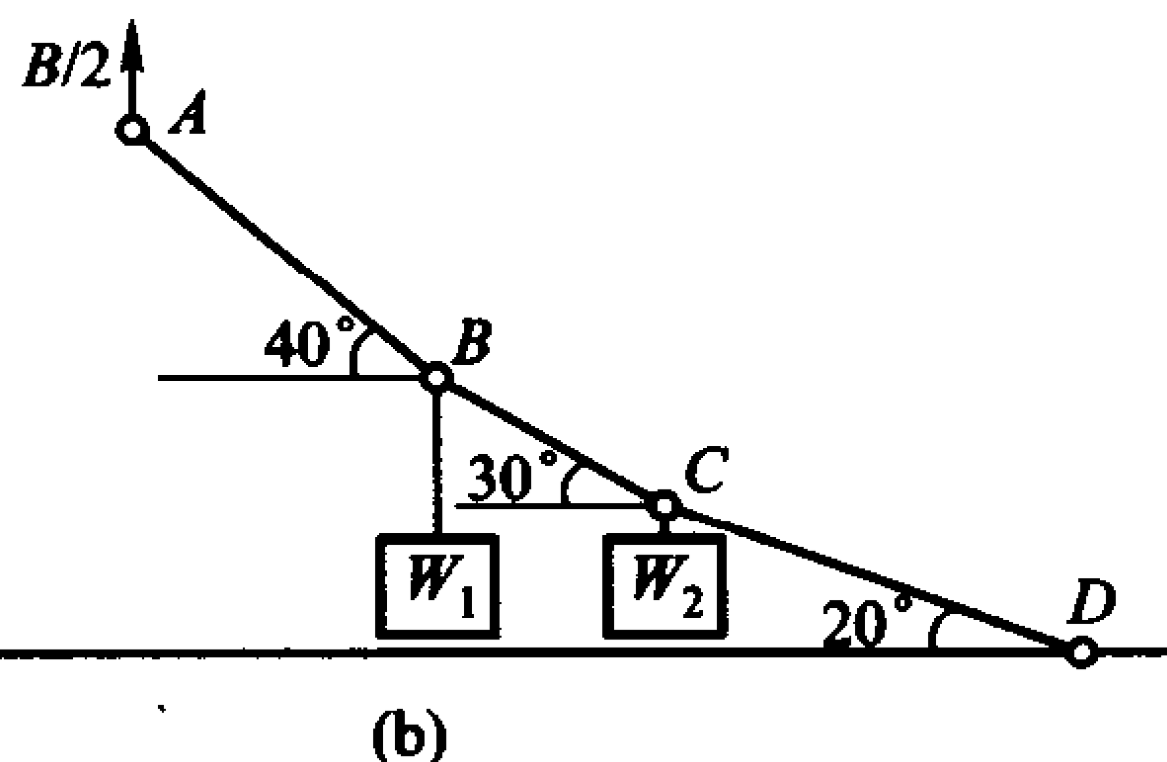


(b)

图 4－3



(a)



(b)

图 4－4

5. 表量程 0～1 mA,内阻 55 Ω,现用分流器来扩大量程如图 4－5 所示,当转臂转至图中

A,B,C,D 位置时，要求其量程分别为 0 ~ 10，0 ~ 50，0 ~ 100，0 ~ 500 (mA)，确定分流器中各电阻值.

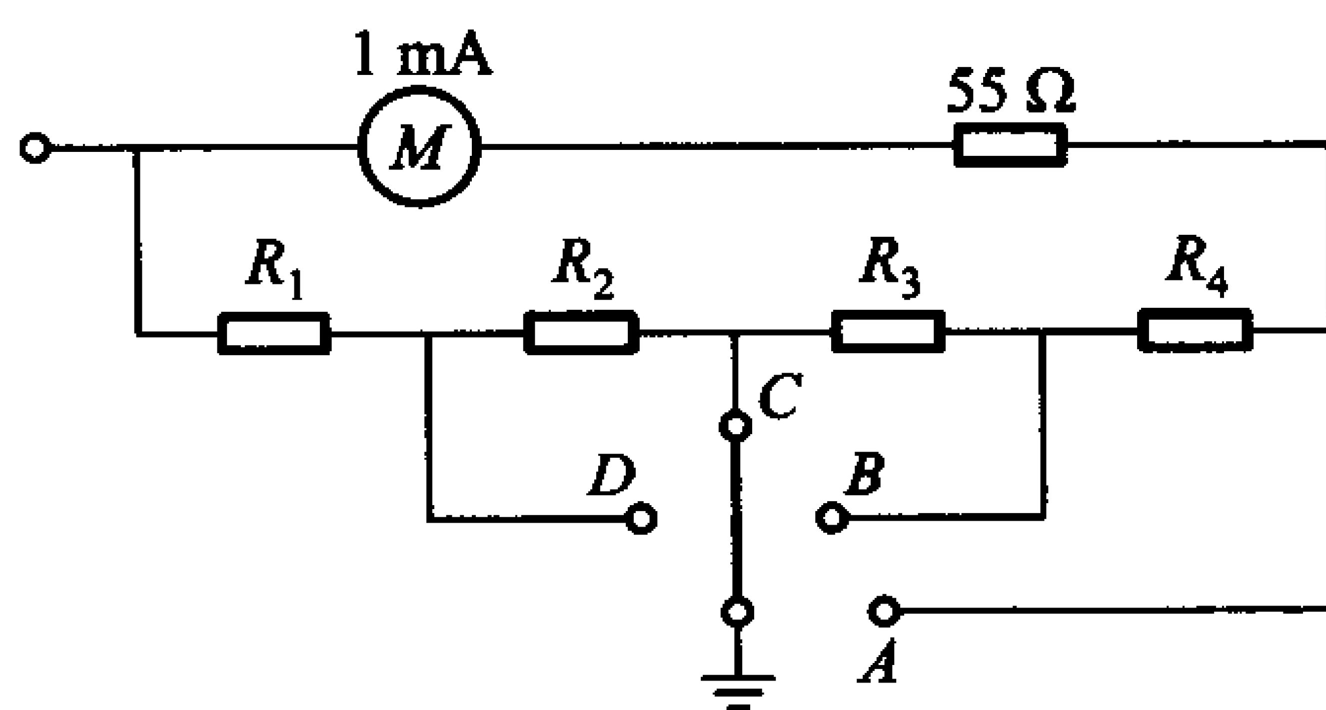


图 4 - 5

6. 种群的数量因繁殖而增加，因自然死亡而减少，对于人工饲养的种群（比如家畜）而言，为了保证稳定的收获，各个年龄的种群数量应维持不变. 种群因雌性个体的繁殖而改变，为方便起见以下种群数量均指其中的雌性.

种群年龄记作 $k=1,2,\cdots,n$，当年龄 k 的种群数量记作 x_k，繁殖率记作 b_k（每个雌性个体一年繁殖的数量），自然存活率记作 s_k （$s_k=1-d_k$，d_k 为一年的死亡率），收获量记作 h_k，则来年年龄 k 的种群数量 $\tilde{x}_k$ 应为 $\tilde{x}_1 = \sum\limits_{k=1}^{n} b_k x_k$，$\tilde{x}_{k+1} = s_k x_k - h_k$ （$k=1,2,\cdots,n-1$）. 要求各个年龄的种群数量每年维持不变就是要使 $\tilde{x}_k = x_k$ （$k=1,2,\cdots,n$）.

(1) 若 b_k, s_k 已知，给定收获量 h_k，建立求各年龄的稳定种群数量 x_k 的模型（用矩阵、向量表示）.

(2) 设 $n=5$，$b_1=b_2=b_5=0$，$b_3=5$，$b_4=3$，$s_1=s_4=0.4$，$s_2=s_3=0.6$，如要求 h_k （$k=1,2,\cdots,5$）为 500，400，200，100，100，求 x_k （$k=1,2,\cdots,5$）.

(3) 欲使 h_k （$k=1,2,\cdots,5$）均为 500，如何达到.

第五章 矩阵的特征值与特征向量的计算

在自然科学和工程设计中的许多问题,如电磁振荡、桥梁振动、机械振动等,常归结为求矩阵的特征值和特征向量.求矩阵的特征值和特征向量的问题是代数计算中的重要课题.本章着重介绍直接计算矩阵的特征值和特征向量的MATLAB程序、间接计算矩阵的特征值和特征向量的幂法、反幂法、雅可比方法、豪斯霍尔德方法和QR方法及其MATLAB计算程序.最后我们还讨论广义特征值问题.

5.1 直接计算特征值和特征向量的MATLAB程序

本节主要介绍矩阵的特征值与特征向量的有关概念和性质,直接计算特征值与特征向量的MATLAB函数的使用方法.

5.1.1 矩阵的特征值和特征向量

定义 5.1 设矩阵 $\boldsymbol{A}=(a_{ij})_{n\times n}\in\mathbf{C}^{n\times n}$,如果复数 λ 和 n 维非零列向量 $\boldsymbol{X}$ 使关系式

$$\boldsymbol{AX}=\lambda\boldsymbol{X} \tag{5.1}$$

成立,则这个复数 λ 称为方阵 $\boldsymbol{A}$ 的特征值,非零列向量 $\boldsymbol{X}$ 称为 $\boldsymbol{A}$ 的对应于特征值 λ 的特征向量.

n 阶方阵 $\boldsymbol{A}$ 的特征值就是使齐次线性方程组

$$(\boldsymbol{A}-\lambda\boldsymbol{E})\boldsymbol{X}=\boldsymbol{O} \tag{5.2}$$

有非零解的 λ 值,即满足特征方程 $\det(\boldsymbol{A}-\lambda\boldsymbol{E})=0$ 的 λ 都是方阵 $\boldsymbol{A}$ 的特征值.

特征方程 $\det(\boldsymbol{A}-\lambda\boldsymbol{E})=0$ 可以表示为如下形式:

$$\begin{vmatrix} a_{11}-\lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22}-\lambda & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn}-\lambda \end{vmatrix}=0. \tag{5.3}$$

定义 5.2 设有一组列(行)向量 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n$,如果对于任意 $\boldsymbol{X}_j\neq\boldsymbol{X}_k(j\neq k,\ j,k=1,2,\cdots,n)$ 都有

$$(\boldsymbol{X}_j,\boldsymbol{X}_k)=\boldsymbol{X}_j^{\mathrm{T}}\boldsymbol{X}_k=0 \qquad ((\boldsymbol{X}_j,\boldsymbol{X}_k)=\boldsymbol{X}_j\boldsymbol{X}_k^{\mathrm{T}}=0),$$

则称这组向量是**两两正交的**.

求矩阵的特征值与特征向量的步骤如下：

步骤1 计算 A 的特征多项式 $\det(A-\lambda E)$；

步骤2 求特征方程 $\det(A-\lambda E)=0$ 的全部根 $\lambda_1,\lambda_2,\cdots,\lambda_n$，即是 A 的全部特征值；

步骤3 对于特征值 λ_i，求齐次线性方程组

$$(A-\lambda_i E)X=O$$

的非零解 X，即为 A 的对应于特征值 λ_i 的特征向量.

例 5.1.1 求下列矩阵的特征值和特征向量，并判别它们的特征向量是否两两正交？

(1) $A=\begin{pmatrix}1 & -1\\ 2 & 4\end{pmatrix}$； (2) $B=\begin{pmatrix}1 & 2 & 3\\ 2 & 1 & 3\\ 3 & 3 & 6\end{pmatrix}$.

解 (1) ① $\det(A-\lambda E)=\begin{vmatrix}1-\lambda & -1\\ 2 & 4-\lambda\end{vmatrix}=(\lambda-2)(\lambda-3)=0$,

故 A 的特征值为 $\lambda_1=2,\lambda_2=3$.

② 当 $\lambda_1=2$ 时，解方程 $(A-2E)X=O$，由

$$A-2E=\begin{pmatrix}-1 & -1\\ 2 & 2\end{pmatrix}\sim\begin{pmatrix}1 & 1\\ 0 & 0\end{pmatrix}\text{，得基础解系 } X_1=(-1,1)^{\mathrm{T}},$$

所以 $k_1X_1(k_1\neq 0)$ 是对应于 $\lambda_1=2$ 的全部特征向量.

当 $\lambda_2=3$ 时，解方程 $(A-3E)X=O$，由

$$A-3E=\begin{pmatrix}-2 & -1\\ 2 & 1\end{pmatrix}\sim\begin{pmatrix}2 & 1\\ 0 & 0\end{pmatrix}\text{，得基础解系 } X_2=\left(\frac{1}{2},-1\right)^{\mathrm{T}},$$

所以 $k_2X_2(k_2\neq 0)$ 是对应于 $\lambda_2=3$ 的全部特征向量.

③ $(X_1,X_2)=X_1^{\mathrm{T}}X_2=(-1,1)\begin{pmatrix}\frac{1}{2}\\ -1\end{pmatrix}=-\frac{3}{2}\neq 0$，故 X_1,X_2 不正交.

(2) ① $\det(A-\lambda E)=\begin{vmatrix}1-\lambda & 2 & 3\\ 2 & 1-\lambda & 3\\ 3 & 3 & 6-\lambda\end{vmatrix}=-\lambda(\lambda+1)(\lambda-9)=0$,

故 A 的特征值为 $\lambda_1=0,\lambda_2=-1,\lambda_3=9$.

② 当 $\lambda_1=0$ 时，解方程 $AX=O$，由

$$A=\begin{pmatrix}1 & 2 & 3\\ 2 & 1 & 3\\ 3 & 3 & 6\end{pmatrix}\sim\begin{pmatrix}1 & 2 & 3\\ 0 & 1 & 1\\ 0 & 0 & 0\end{pmatrix}\sim\begin{pmatrix}1 & 0 & 1\\ 0 & 1 & 1\\ 0 & 0 & 0\end{pmatrix}\text{，得基础解系 } X_1=\begin{pmatrix}-1\\ -1\\ 1\end{pmatrix},$$

故 $k_1X_1(k_1\neq 0)$是对应于 $\lambda_1=0$ 的全部特征向量.

当 $\lambda_2=-1$ 时,解方程$(\boldsymbol{A}+\boldsymbol{E})\boldsymbol{X}=\boldsymbol{O}$,由

$$\boldsymbol{A}+\boldsymbol{E}=\begin{pmatrix}2&2&3\\2&2&3\\3&3&7\end{pmatrix}\sim\begin{pmatrix}2&2&3\\0&0&1\\0&0&0\end{pmatrix}\sim\begin{pmatrix}1&1&0\\0&0&1\\0&0&0\end{pmatrix},\text{得基础解系 } \boldsymbol{X}_2=\begin{pmatrix}-1\\1\\0\end{pmatrix},$$

故 $k_2X_2(k_2\neq 0)$是对应于 $\lambda_2=-1$ 的全部特征向量.

当 $\lambda_3=9$ 时,解方程$(\boldsymbol{A}-9\boldsymbol{E})\boldsymbol{X}=\boldsymbol{O}$,由

$$\boldsymbol{A}-9\boldsymbol{E}=\begin{pmatrix}-8&2&3\\2&-8&3\\3&3&-3\end{pmatrix}\sim\begin{pmatrix}1&1&-1\\0&-2&1\\0&0&0\end{pmatrix}\sim\begin{pmatrix}1&-1&0\\0&-2&1\\0&0&0\end{pmatrix},$$

得基础解系 $\boldsymbol{X}_3=\begin{pmatrix}1\\1\\2\end{pmatrix}$,故 $k_3X_3(k_3\neq 0)$是对应于 $\lambda_3=9$ 的全部特征向量.

③ $(\boldsymbol{X}_1,\boldsymbol{X}_2)=\boldsymbol{X}_1^{\mathrm{T}}\boldsymbol{X}_2=(-1,-1,1)(-1,1,0)^{\mathrm{T}}=0$,

$(\boldsymbol{X}_2,\boldsymbol{X}_3)=\boldsymbol{X}_2^{\mathrm{T}}\boldsymbol{X}_3=(-1,1,0)(1,1,2)^{\mathrm{T}}=0$,

$(\boldsymbol{X}_1,\boldsymbol{X}_3)=\boldsymbol{X}_1^{\mathrm{T}}\boldsymbol{X}_3=(-1,-1,1)(1,1,2)^{\mathrm{T}}=0$,

所以 $\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{X}_3$ 两两正交.

定理 5.1 设 $\boldsymbol{A}$ 是 $n\times n$ 阶矩阵,则

(1) 对于 $\boldsymbol{A}$ 的每个特征值 λ_i 至少有一个 $\boldsymbol{A}$ 的对应于特征值 λ_i 的特征向量 $\boldsymbol{X}_{it}$;

(2) 如果 λ_i 是 $\boldsymbol{A}$ 的 r 重特征值,则 $\boldsymbol{A}$ 的对应于特征值 λ_i 的所有线性无关的特征向量 $\boldsymbol{X}_{i1},\boldsymbol{X}_{i2},\cdots,\boldsymbol{X}_{it}$至多有 r 个,即 $t\leqslant r$.

定理 5.2 设 $\boldsymbol{A}$ 是 $n\times n$ 阶矩阵,$\lambda_1,\lambda_2,\cdots,\lambda_k(k\leqslant n)$是 $\boldsymbol{A}$ 的互不相同的特征值,对应的特征向量分别为 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_k$,则 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_k$ 线性无关.

定理 5.3 如果 $n\times n$ 阶矩阵 $\boldsymbol{A}$ 的所有特征值 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 是互不相同的,则存在 n 个线性无关的特征向量 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n$.

例如,例 5.1.1(1)中的 2 阶矩阵 $\boldsymbol{A}$ 的特征值为 $\lambda_1=2\neq\lambda_2=3$,则存在两个线性无关的特征向量 $\boldsymbol{X}_1=(-1,1)^{\mathrm{T}},\boldsymbol{X}_2=(1,-2)^{\mathrm{T}}$.(2)也有类似的结果.

矩阵的特征值与特征向量有如下性质:

定理 5.4 设 n 阶方阵 $\boldsymbol{A}=(a_{ij})$的全部特征值为 $\lambda_1,\lambda_2,\cdots,\lambda_n$,则有

(1) $\lambda_1+\lambda_2+\cdots+\lambda_n=a_{11}+a_{22}+\cdots+a_{nn}$;

(2) $\lambda_1\lambda_2\cdots\lambda_n=\det\boldsymbol{A}$;

(3) 属于同一个特征值 λ_i 的特征向量 $\boldsymbol{X}_{i1},\boldsymbol{X}_{i2},\cdots,\boldsymbol{X}_{it}$的非零线性组合

$\sum_{r=1}^{t} k_r X_{ir}$（其中 $k_1,k_2,\cdots,k_t$ 是不同时为零的任意常数）仍是属于这个特征值 λ_i 的特征向量.

定理 5.5 设 λ 是 $A=(a_{ij})_{n\times n}$ 的特征值，则

(1) λ 也是 A^T 的特征值；

(2) λ^k 是 A^k 的特征值（k 为任意实数）；

(3) $\varphi(\lambda)$ 是 $\varphi(A)$ 的特征值，其中

$$\varphi(\lambda)=a_0+a_1\lambda+\cdots+a_m\lambda^m,\varphi(A)=a_0E+a_1A+\cdots+a_mA^m;$$

(4) 当 A 可逆时，$\frac{1}{\lambda}$ 是 A^{-1} 的特征值，$\frac{1}{\lambda}\det A$ 是 A^* 的特征值.

定理 5.6（格什戈林（Gerschgorin）圆盘定理） 设矩阵 $A=(a_{ij})_{n\times n}$，则 A 的每一个特征值 λ 必属于下述某个圆盘之中：

$$|\lambda-a_{ii}|\leqslant\sum_{\substack{j=1\\j\neq i}}^{n}|a_{ij}|\quad(i=1,2,\cdots,n).$$

5.1.2 矩阵的对角化

定义 5.3 设 A,B 都是 n 阶矩阵，如果存在可逆矩阵 P，使 $P^{-1}AP=B$，则称 B 是 A 的**相似矩阵**，可逆矩阵 P 称为把 A 变成 B 的**相似变换矩阵**.

定理 5.7 若 n 阶方阵 A 与对角矩阵

$$\Lambda=\begin{pmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n\end{pmatrix}$$

相似，则 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 是 A 的 n 个特征值.

定理 5.8 如果 n 阶矩阵 A 与 B 相似，则 A 与 B 的特征多项式相同.

由定理 5.8 知，如果 n 阶矩阵 A 与 B 相似，则 A 与 B 有相同的特征值.

定义 5.4 对于 n 阶方阵 A，如果存在可逆矩阵 P，使 $P^{-1}AP=\Lambda$ 为对角矩阵，这就称为把**方阵 A 对角化**.

定理 5.9 n 阶方阵 A 与对角矩阵相似，即 A 能对角化的充分必要条件是 A 有 n 个线性无关的特征向量.

推论 5.1 如果 n 阶矩阵 A 的 n 个特征值互不相等，则 A 与对角矩阵相似.

如果 n 阶矩阵 A 的特征方程有重根，此时不一定有 n 个线性无关的特征向量，从而矩阵 A 不一定能对角化，但如果能找到 n 个线性无关的特征向量，A 还是能对角化的.

例 5.1.2 判断例 5.1.1 中的矩阵能否化为对角矩阵？

解 (1) 根据例 5.1.1 的计算结果，知 2 阶矩阵 $\boldsymbol{A}$ 有两个不同的特征值 $\lambda_1=2,\lambda_2=3$，由推论 5.1 知矩阵 $\boldsymbol{A}$ 可以对角化. 同理可知(2)中的矩阵 $\boldsymbol{A}$ 可以对角化.

定义 5.5 如果 n 阶方阵 $\boldsymbol{A}$ 的转置矩阵 $\boldsymbol{A}^{\mathrm{T}}$ 等于逆矩阵 $\boldsymbol{A}^{-1}$，则称方阵 $\boldsymbol{A}$ 为**正交矩阵**.

由定义 5.5 知，n 阶方阵 $\boldsymbol{A}$ 为**正交矩阵的充分必要条件**是 $\boldsymbol{A}$ 的 n 个列 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n$（或行）向量都是单位向量，且两两正交，即

$$\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{b}_j=\begin{cases}1, i=j,\\0, i\neq j\end{cases}\qquad (i,j=1,2,\cdots,n).$$

5.1.3 实对称矩阵

实对称矩阵有四个性质如下：

定理 5.10 实对称矩阵的特征值为实数.

定理 5.10 的意义是：由于实对称矩阵 $\boldsymbol{A}$ 的特征值 λ_i 为实数，所以齐次线性方程组 $(\boldsymbol{A}-\lambda_i\boldsymbol{E})\boldsymbol{X}=\boldsymbol{O}$ 是实系数矩阵方程组. 由 $\det(\boldsymbol{A}-\lambda_i\boldsymbol{E})=0$ 知，必有实的基础解系，从而对应的特征向量可以取实向量.

定理 5.11 设 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 是 n 阶实对称矩阵 $\boldsymbol{A}$ 的特征值，满足 $\lambda_1\geqslant\lambda_2\geqslant\cdots\geqslant\lambda_n$，对应的特征向量组 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n$ 满足 $(\boldsymbol{X}_i,\boldsymbol{X}_j)=\begin{cases}1, i=j,\\0, i\neq j\end{cases}$ $(i,j=1,2,\cdots,n)$，则对于任意 n 维非零向量 $\boldsymbol{X}$ 有以下结论：

$$(1)\ \lambda_1\geqslant\frac{(\boldsymbol{AX},\boldsymbol{X})}{(\boldsymbol{X},\boldsymbol{X})}\geqslant\lambda_n;(2)\ \lambda_1=\max_{\substack{\boldsymbol{X}\in\mathbf{R}^n\\\boldsymbol{X}\neq\boldsymbol{0}}}\frac{(\boldsymbol{AX},\boldsymbol{X})}{(\boldsymbol{X},\boldsymbol{X})};(3)\ \lambda_n=\min_{\substack{\boldsymbol{X}\in\mathbf{R}^n\\\boldsymbol{X}\neq\boldsymbol{0}}}\frac{(\boldsymbol{AX},\boldsymbol{X})}{(\boldsymbol{X},\boldsymbol{X})}.$$

定理 5.12 设 λ_1,λ_2 是实对称矩阵 $\boldsymbol{A}$ 的特征值，$\boldsymbol{X}_1,\boldsymbol{X}_2$ 分别是 λ_1,λ_2 对应的特征向量，如果 $\lambda_1\neq\lambda_2$，则 $\boldsymbol{X}_1$ 与 $\boldsymbol{X}_2$ 正交.

定理 5.13 如果 λ 是 n 阶实对称矩阵 $\boldsymbol{A}$ 的 r 重特征值，则矩阵 $\boldsymbol{A}-\lambda\boldsymbol{E}$ 的秩 $R(\boldsymbol{A}-\lambda\boldsymbol{E})=n-r$，$\boldsymbol{A}$ 的特征值 λ 恰有 r 个线性无关的特征向量.

定理 5.14 如果 $\boldsymbol{A}$ 是 n 阶实对称矩阵，则必有正交矩阵 $\boldsymbol{P}$，使 $\boldsymbol{P}^{-1}\boldsymbol{AP}=\boldsymbol{\Lambda}$，其中 $\boldsymbol{\Lambda}$ 是以 $\boldsymbol{A}$ 的 n 个特征值为对角元的对角矩阵.

定理 5.15 实对称矩阵 $\boldsymbol{A}$ 为正定的充分必要条件是 $\boldsymbol{A}$ 的特征值全为正.

根据上述结论，利用正交矩阵将实对称矩阵 $\boldsymbol{A}$ 化为对角矩阵的一般步骤为：求特征方程 $\det(\boldsymbol{A}-\lambda\boldsymbol{E})=0$ 的全部根 $\lambda_1,\lambda_2,\cdots,\lambda_n$，即为 $\boldsymbol{A}$ 的全部特征值；对于特征值 λ_i，求齐次线性方程组 $(\boldsymbol{A}-\lambda_i\boldsymbol{E})\boldsymbol{X}=\boldsymbol{O}$ 的基础解系，即为 $\boldsymbol{A}$ 的对应于特征值 λ_i 的特征向量. 将所得到的特征向量正交化，然后单位化，将其作为列向量构造正交矩阵 $\boldsymbol{P}$，使得 $\boldsymbol{P}^{-1}\boldsymbol{AP}=\boldsymbol{\Lambda}$ 为对角矩阵.

例 5.1.3 试求一个正交的相似变换矩阵,将下列对称矩阵化为对角矩阵:

$$\begin{pmatrix} 2 & 2 & -2 \\ 2 & 5 & -4 \\ -2 & -4 & 5 \end{pmatrix}.$$

解 $\det(\boldsymbol{A}-\lambda\boldsymbol{E})=\begin{vmatrix} 2-\lambda & 2 & -2 \\ 2 & 5-\lambda & -4 \\ -2 & -4 & 5-\lambda \end{vmatrix}=-(\lambda-1)^2(\lambda-10)=0,$

解得特征值为 $\lambda_1=\lambda_2=1,\lambda_3=10$.

当 $\lambda_1=\lambda_2=1$ 时,由 $\begin{pmatrix} 1 & 2 & -2 \\ 2 & 4 & -4 \\ -2 & -4 & 4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$,解得基础解系

$$\boldsymbol{\xi}_1=\begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}, \boldsymbol{\xi}_2=\begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}.$$

将此两个向量正交化,单位化后,得两个单位正交的特征向量

$$\boldsymbol{P}_1=\frac{1}{\sqrt{5}}\begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{P}_2=\frac{\sqrt{5}}{3}\begin{pmatrix} 2/5 \\ 4/5 \\ 1 \end{pmatrix}.$$

当 $\lambda_3=10$ 时,由 $\begin{pmatrix} -8 & 2 & -2 \\ 2 & -5 & -4 \\ -2 & -4 & -5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$,解得基础解系 $\boldsymbol{\xi}_3=\begin{pmatrix} -1 \\ -2 \\ 2 \end{pmatrix}$,单位化得

$$\boldsymbol{P}_3=\frac{1}{3}\begin{pmatrix} -1 \\ -2 \\ 2 \end{pmatrix},$$

可以得正交矩阵 $\boldsymbol{P}=(\boldsymbol{P}_1,\boldsymbol{P}_2,\boldsymbol{P}_3)=\begin{pmatrix} -\frac{2}{\sqrt{5}} & \frac{2\sqrt{5}}{15} & -\frac{1}{3} \\ \frac{1}{\sqrt{5}} & \frac{4\sqrt{5}}{15} & -\frac{2}{3} \\ 0 & \frac{\sqrt{5}}{3} & \frac{2}{3} \end{pmatrix}$,

使得 $\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10 \end{pmatrix}$.

5.1.4 计算特征值和特征向量的 MATLAB 程序

从以上的讨论可以看到,有许多问题归结为求矩阵的特征值和特征向量,而用手工计算高阶矩阵的特征值与特征向量的难度较大,但是,计算机软件 MATLAB 提供了直接计算特征值与特征向量的 MATLAB 函数(见表 5-1),下面介绍这些函数的使用方法.

表 5-1

命令	功能
b=eig(A)	输入方阵 $\boldsymbol{A}$,运行后输出 $\boldsymbol{b}$ 为由方阵 $\boldsymbol{A}$ 的全部特征值构成的列向量
[V,D]=eig(A)	输入对称矩阵 $\boldsymbol{A}$,运行后输出 $\boldsymbol{D}$ 为由 $\boldsymbol{A}$ 的全部特征值构成的对角矩阵,$\boldsymbol{V}$ 的各列为对应于特征值的特征向量构成的矩阵,使得 $\boldsymbol{AV}=\boldsymbol{DV}$
[V,D]=eig(A,'nobalance')	输入方阵 $\boldsymbol{A}$,运行后输出 $\boldsymbol{D}$ 为由 $\boldsymbol{A}$ 的全部特征值构成的对角矩阵,$\boldsymbol{V}$ 的各列为对应于特征值的特征向量构成的矩阵,使得 $\boldsymbol{AV}=\boldsymbol{DV}$;如果 $\boldsymbol{A}$ 是对称矩阵,则输出的结果与程序[V,D]=eig(A)的运行结果相同

例 5.1.4 用 MATLAB 程序求下列矩阵的特征值和特征向量,并把(1)和(2)输出的结果与例 5.1.1 中的结果进行比较.

$$(1)\ \boldsymbol{A}=\begin{pmatrix}1 & -1\\ 2 & 4\end{pmatrix};(2)\ \boldsymbol{B}=\begin{pmatrix}1 & 2 & 3\\ 2 & 1 & 3\\ 3 & 3 & 6\end{pmatrix};(3)\ \boldsymbol{C}=\begin{pmatrix}1 & 2 & 2\\ 1 & -1 & 1\\ 4 & -12 & 1\end{pmatrix}.$$

解 (1) 输入 MATLAB 程序

```
>>A=[1 -1;2  4];[V,D]=eig(A,'nobalance'),
```

运行后输出结果

```
V=-1.0000    0.5000            D=2  0
   1.0000   -1.0000              0  3
```

输入 MATLAB 程序

```
>>b=eig(A);b'
```

运行后输出结果

```
ans=2      3
```

输入 MATLAB 程序

```
>>[V,D]=eig(A)
```

运行后输出结果

```
V = -0.70710678118655    0.44721359549996        D = 2   0
     0.70710678118655   -0.89442719099992            0   3
```

可见,方阵 A 不是对称矩阵,用程序 eig(A,'nobalance')输出的结果与例 5.1.1 中 A 的特征值 $\lambda_1=2,\lambda_2=3$ 和特征向量 $X_1=k_1(-1,1)^T,X_2=k_2\left(\frac{1}{2},-1\right)^T(k_1\neq0,k_2\neq0)$ 相同. 用程序 eig(A)输出的结果与例 5.1.1 中 A 的特征值相同,特征向量形式上不同,但也是 A 的特征向量.

(2) 输入 MATLAB 程序

```
>>B=[1  2  3;2  1  3;3  3  6];[V,D]=eig(B,'nobalance')
```

运行后输出结果

```
V =                                 D =
   0.7071    0.5774   0.4082           -1.0000         0         0
  -0.7071    0.5774   0.4082                 0   -0.0000         0
        0   -0.5774   0.8165                 0         0    9.0000
```

输入 MATLAB 程序

```
>>E=eig(B);E'
```

运行后输出结果

```
E' = -1.0000     -0.0000     9.0000
```

输入 MATLAB 程序

```
>>[V,D]=eig(B)
```

运行后输出结果

```
V =                                 D =
   0.7071    0.5774   0.4082           -1.0000         0         0
  -0.7071    0.5774   0.4082                 0   -0.0000         0
        0   -0.5774   0.8165                 0         0    9.0000
```

可见,方阵 A 是对称矩阵,用程序 eig(A,'nobalance')和程序 eig(A)输出的结果相同. 这些结果与例 5.1.1 中 A 的特征值 $\lambda_1=0,\lambda_2=-1,\lambda_3=9$ 和特征向量 $X_1=k_1(-1,-1,1)^T,X_2=k_2(-1,1,0)^T,X_3=k_3(1,1,2)^T(k_1\neq0,k_2\neq0,k_3\neq0)$ 比较,特征值相同,特征向量形式上不同,但也是 A 的特征向量.

(3) 输入 MATLAB 程序

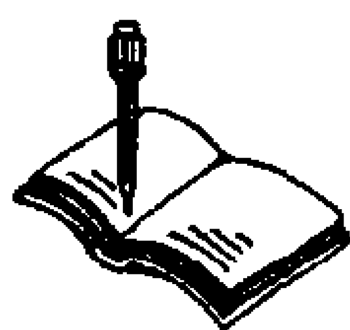
```
>>C=[1  2  2;1  -1  1;4 -12  1];[V,D]=eig(C,'nobalance')
```

运行后输出结果

```
V =
  -1.0000   -0.5280 -0.4720i   -0.5280 +0.4720i
  -0.3333   -0.1112  0.1944i   -0.1112 +0.1944i
   0.3333    0.6112 +0.1664i    0.6112 -0.1664i
```

```
D =
   1.0000                 0                    0
        0   -0.0000 +1.0000i                   0
        0                 0     -0.0000 -1.0000i
```

虽然矩阵 C 中的元全部是实数,但是 C 的特征值和特征向量却是复数.



习 题 5.1

1. 试用 MATLAB 程序计算下列矩阵的特征值和特征向量.

(1) $\begin{pmatrix} -5 & 2 & 17 \\ 2 & -8 & 3 \\ 3 & 3 & -3 \end{pmatrix}$;(2) $\begin{pmatrix} -11 & 2 & 15 \\ 2 & 58 & 3 \\ 15 & 3 & -3 \end{pmatrix}$;(3) $\begin{pmatrix} -2 & 1 & -2 \\ -5 & 3 & -3 \\ 1 & 0 & 2 \end{pmatrix}$;

(4) $\begin{pmatrix} 4 & 6 & 0 \\ -3 & -5 & 0 \\ -3 & -6 & 1 \end{pmatrix}$.

2. 试用 MATLAB 程序求一个正交的相似变换矩阵,将下列对称矩阵化为对角矩阵:

(1) $\boldsymbol{A}=\begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 1 \\ -1 & 0 & 1 & 1 \end{pmatrix}$;(2) $\boldsymbol{B}=\begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 2 \\ 0 & 2 & 3 \end{pmatrix}$.

5.2 幂法及其 MATLAB 程序

幂法是求实矩阵 $\boldsymbol{A}$ 的主特征值(即实矩阵 $\boldsymbol{A}$ 按模最大的特征值)及其对应的特征向量的一种迭代方法.本节主要介绍幂法的思想方法、收敛性和用幂法求实矩阵 $\boldsymbol{A}$ 的主特征值和对应的特征向量的 MATLAB 程序.

5.2.1 幂法

定义 5.6 如果 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 是 n 阶实矩阵 $\boldsymbol{A}$ 的特征值,且满足

$$|\lambda_1|>|\lambda_2|\geqslant\cdots\geqslant|\lambda_n|,$$

则称 λ_1 为**主特征值**,对应的向量 $\boldsymbol{X}_1$ 为**主特征向量**,称 λ_n 为**按模最小特征值**.

设 n 阶实矩阵 $\boldsymbol{A}$ 有一组含 n 个线性无关的特征向量组 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n$,它们分别对应于特征值 $\lambda_1,\lambda_2,\cdots,\lambda_n$,满足

$$|\lambda_1|\geqslant|\lambda_2|\geqslant\cdots\geqslant|\lambda_n|. \tag{5.4}$$

从而有

$$AX_i=\lambda_i X_i(i=1,2,\cdots,n).\tag{5.5}$$

首先,我们讨论 λ_1 是实数而且是单重特征值的情形,此时有

$$|\lambda_1|>|\lambda_2|\geqslant\cdots\geqslant|\lambda_n|.\tag{5.6}$$

幂法的基本思想是任取一个非零 n 维初始实向量 V_0,由矩阵 A 构造一个向量序列:$V_k=AV_{k-1}(k=1,2,\cdots)$,则有 $V_k=AV_{k-1}=A^2V_{k-2}=\cdots=A^kV_0$. 因为特征向量组 $X_1,X_2,\cdots,X_n$ 线性无关,所以,n 维初始实向量 V_0 可表示为矩阵 A 的特征向量组的一个线性组合

$$V_0=\sum_{i=1}^{n}a_iX_i.$$

不妨设 $a_1\neq 0$,于是 $V_k=A^kV_0=\sum_{i=1}^{n}a_iA^kX_i=\sum_{i=1}^{n}a_i\lambda_i^kX_i$,得

$$V_k=\lambda_1^k\left[a_1X_1+\sum_{i=2}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^kX_i\right],\tag{5.7}$$

即

$$A^kV_0=\lambda_1^k\left[a_1X_1+\sum_{i=2}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^kX_i\right].\tag{5.8}$$

设 $V_k=\{v_k^{(1)},v_k^{(2)},\cdots,v_k^{(n)}\}$,$X_i=\{x_i^{(1)},x_i^{(2)},\cdots,x_i^{(n)}\}$,由(5.7)式,得

$$\frac{v_{k+1}^{(j)}}{v_k^{(j)}}=\lambda_1\frac{a_1x_1^{(j)}+a_2x_2^{(j)}\left(\frac{\lambda_2}{\lambda_1}\right)^{k+1}+\sum_{i=3}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^{k+1}x_i^{(j)}}{a_1x_1^{(j)}+\sum_{i=2}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^kx_i^{(j)}}.\tag{5.9}$$

因为 $\left|\frac{\lambda_i}{\lambda_1}\right|<1\ (i=2,3,\cdots,n)$,所以 $\lim\limits_{\substack{k\to\infty\\ i=2,3,\cdots,n}}\left(\frac{\lambda_i}{\lambda_1}\right)^k=0$,由(5.9)式得

$$\lim_{k\to\infty}\frac{v_{k+1}^{(j)}}{v_k^{(j)}}=\lambda_1.\tag{5.10}$$

由(5.7)式知

$$\lim_{k\to\infty}\frac{V_k}{\lambda_1^k}=a_1X_1.\tag{5.11}$$

由(5.10)式知,当 k 充分大时,有 $\lambda_1\approx\frac{v_{k+1}^{(j)}}{v_k^{(j)}}$.

这种由已知非零 n 维初始实向量 V_0 和矩阵 A 的乘幂 A^k 构造一个向量序列 $\{V_k\}$,求矩阵 A 的主特征值的方法称作**幂法**. 由(5.10)式可见,当 k 取不同的充分大的数时,$v_{k+1}^{(j)}/v_k^{(j)}$ 也不同,从而 λ_1 的近似值就不同,这时可用 $v_{k+1}^{(j)}/v_k^{(j)}$ 的平均值作为 λ_1 的近似值.

假定 $a_1\neq 0,x_1^{(j)}\neq 0$,由(5.9)式,得

$$\frac{v_{k+1}^{(j)}}{v_k^{(j)}}-\lambda_1=\lambda_1\frac{\frac{a_2x_2^{(j)}}{a_1x_1^{(j)}}\left(\frac{\lambda_2}{\lambda_1}\right)^k\left(\frac{\lambda_2}{\lambda_1}-1\right)+\sum_{i=3}^{n}\frac{a_ix_i^{(j)}}{a_1x_1^{(j)}}\left(\frac{\lambda_i}{\lambda_1}\right)^k\left(\frac{\lambda_i}{\lambda_1}-1\right)}{1+\sum_{i=2}^{n}\frac{a_ix_i^{(j)}}{a_1x_1^{(j)}}\left(\frac{\lambda_i}{\lambda_1}\right)^k}. \quad (5.12)$$

由(5.6),(5.10)和(5.12)式,得

$$\frac{v_{k+1}^{(j)}}{v_k^{(j)}}=\lambda_1+O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^k\right)(k\to\infty). \quad (5.13)$$

由(5.13)式可见,幂法的收敛速度主要取决于$|\lambda_2/\lambda_1|$的大小,$|\lambda_2/\lambda_1|$越接近于1,迭代收敛的速度就越慢.

现在,我们来讨论矩阵 $\boldsymbol{A}$ 与 λ_1 对应的特征向量 $\boldsymbol{X}_1$ 的计算. 由(5.11)式可得,当 k 充分大时,

$$\boldsymbol{V}_k\approx\lambda_1^k a_1\boldsymbol{X}_1, \quad (5.14)$$

即 $\boldsymbol{V}_k$ 可以作为与 λ_1 对应的特征向量 $\boldsymbol{X}_1$ 的近似向量.

定理 5.16 设 n 阶实矩阵 $\boldsymbol{A}$ 有 n 个线性无关的特征向量 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n$,它们分别对应于实特征值 $\lambda_1,\lambda_2,\cdots,\lambda_n$,

(1) 如果主特征值 λ_1 是单重特征值,即$|\lambda_1|>|\lambda_2|\geqslant\cdots\geqslant|\lambda_n|$,则对于任何非零 n 维初始向量 $\boldsymbol{V}_0$,(5.10)式和(5.11)成立.

(2) 如果矩阵 $\boldsymbol{A}$ 的主特征值 λ_1 为 r 重特征值,即

$$\lambda_1=\lambda_2=\cdots=\lambda_r,\text{且}|\lambda_r|>|\lambda_{r+1}|\geqslant\cdots\geqslant|\lambda_n|,\sum_{i=1}^{r}a_i\boldsymbol{X}_i\neq\boldsymbol{O},$$

则对于任何非零 n 维初始向量 $\boldsymbol{V}_0$,有

$$\lim_{k\to\infty}\frac{\boldsymbol{V}_k}{\lambda_1^k}=\sum_{i=1}^{r}a_i\boldsymbol{X}_i.$$

然而,由(5.7)式或(5.14)式可知,当 $k\to\infty$ 时,如果$|\lambda_1|>1$,则 $\boldsymbol{V}_k$ 的分量 $v_k^{(j)}\to\infty$;如果$|\lambda_1|<1$,则 $\boldsymbol{V}_k$ 的分量 $v_k^{(j)}\to 0$,从而会使计算机出现上溢或下溢的现象. 因此,为了控制计算过程中出现的这种现象,常在每一步中将 $\boldsymbol{V}_k$ 规格化(或规范化),也就是用

$$\begin{cases}\boldsymbol{U}_k=\boldsymbol{A}\boldsymbol{V}_{k-1},\\ \boldsymbol{V}_k=\dfrac{\boldsymbol{U}_k}{m_k}\end{cases}\quad(k=1,2,\cdots). \quad (5.15)$$

其中 $m_k=\max\limits_{1\leqslant i\leqslant n}\{|u_k^{(i)}|\},\boldsymbol{U}_k=\{u_k^{(1)},u_k^{(2)},\cdots,u_k^{(n)}\}\quad(k=1,2,\cdots,n)$.

用(5.15)式代替(5.7)式,由(5.14)式,可得

$$\boldsymbol{V}_k=\frac{\boldsymbol{U}_k}{m_k}=\frac{\boldsymbol{A}\boldsymbol{V}_{k-1}}{m_k}=\frac{\boldsymbol{A}\boldsymbol{U}_{k-1}}{m_k m_{k-1}}=\frac{\boldsymbol{A}^2\boldsymbol{V}_{k-1}}{m_k m_{k-1}}=\cdots=\frac{\boldsymbol{A}^k\boldsymbol{V}_0}{m_k m_{k-1}\cdots m_1}.$$

由 $m_t=\max\limits_{1\leqslant i\leqslant n}\{|u_t^{(i)}|\}=\max(\boldsymbol{U}_t)(t=1,2,\cdots)$ 和(5.15)式,得

$$\max(\boldsymbol{A}^k\boldsymbol{V}_0)=\max(\boldsymbol{A}^{k-1}\boldsymbol{A}\boldsymbol{V}_0)=\max(\boldsymbol{A}^{k-1}\boldsymbol{U}_1)=\max\left(\boldsymbol{A}^{k-1}\frac{\boldsymbol{U}_1}{m_1}m_1\right)$$

$$=\max(\boldsymbol{A}^{k-1}\boldsymbol{V}_1)m_1=\max\left(\boldsymbol{A}^{k-2}\frac{\boldsymbol{U}_2}{m_2}m_2\right)m_1=\cdots=m_km_{k-1}\cdots m_1.$$

因此,

$$\boldsymbol{V}_k=\frac{\boldsymbol{A}^k\boldsymbol{V}_0}{\max(\boldsymbol{A}^k\boldsymbol{V}_0)}. \tag{5.16}$$

将(5.8)式代替(5.16)式,得

$$\boldsymbol{V}_k=\frac{a_1\boldsymbol{X}_1+\sum\limits_{i=2}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\boldsymbol{X}_i}{\max\left(a_1\boldsymbol{X}_1+\sum\limits_{i=2}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\boldsymbol{X}_i\right)}\to\frac{\boldsymbol{X}_1}{\max(\boldsymbol{X}_1)}\qquad(k\to\infty). \tag{5.17}$$

因为特征向量乘以非零常数仍是与原特征值对应的特征向量,所以当 k 充分大时,$\boldsymbol{V}_k$ 是特征向量 $\boldsymbol{X}_1$ 的近似向量.

由(5.15)式和(5.17)式,可得

$$\boldsymbol{U}_k=\boldsymbol{A}\boldsymbol{V}_{k-1}=\frac{a_1\boldsymbol{A}\boldsymbol{X}_1+\sum\limits_{i=2}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^{k-1}\boldsymbol{A}\boldsymbol{X}_i}{\max\left(a_1\boldsymbol{X}_1+\sum\limits_{i=2}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^{k-1}\boldsymbol{X}_i\right)}. \tag{5.18}$$

将(5.5)式代入(5.18)式,得

$$\boldsymbol{U}_k=\lambda_1\frac{a_1\boldsymbol{X}_1+\sum\limits_{i=2}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\boldsymbol{X}_i}{\max\left(a_1\boldsymbol{X}_1+\sum\limits_{i=2}^{n}a_i\left(\frac{\lambda_i}{\lambda_1}\right)^{k-1}\boldsymbol{X}_i\right)}.$$

当 $k\to\infty$ 时,有

$$m_k=\max(\boldsymbol{U}_k)\to\lambda_1. \tag{5.19}$$

因此,当 k 充分大时,

$$m_k\approx\lambda_1.$$

综上所述,有下面定理.

定理 5.17 设 n 阶实矩阵 $\boldsymbol{A}$ 有 n 个线性无关的特征向量 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n$,$\boldsymbol{A}$ 的主特征值 λ_1 满足 $|\lambda_1|>|\lambda_2|\geqslant\cdots\geqslant|\lambda_n|$,则对于任何非零 n 维初始向量 $\boldsymbol{V}_0=\boldsymbol{U}_0$,由幂法的迭代公式

$$\begin{cases} \boldsymbol{U}_k = \boldsymbol{A}\boldsymbol{V}_{k-1}, \\ m_k = \max(\boldsymbol{U}_k), \\ \boldsymbol{V}_k = \dfrac{\boldsymbol{U}_k}{m_k} \end{cases} \quad (k=1,2,\cdots) \tag{5.20}$$

构造的序列$\{\boldsymbol{V}_k\}$和$\{m_k\}$,满足

$$\lim_{k\to\infty}\boldsymbol{V}_k = \frac{\boldsymbol{X}_1}{\max(\boldsymbol{X}_1)} \quad 且 \quad \lim_{k\to\infty}\max(\boldsymbol{U}_k) = \lambda_1.$$

例 5.2.1 用幂法计算矩阵 $\boldsymbol{A} = \begin{pmatrix} -4 & 14 & 0 \\ -5 & 13 & 0 \\ -1 & 0 & 2 \end{pmatrix}$的主特征值和对应的特征向量.

解 取 $\boldsymbol{V}_0 = (1,1,1)^{\mathrm{T}}$,根据迭代公式(5.20),得.

$$\boldsymbol{U}_1 = \boldsymbol{A}\boldsymbol{V}_0 = \begin{pmatrix} -4 & 14 & 0 \\ -5 & 13 & 0 \\ -1 & 0 & 2 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix}, \quad m_1 = \max(\boldsymbol{U}_1) = 10,$$

于是规范化向量,得 $\boldsymbol{V}_1 = \dfrac{\boldsymbol{U}_1}{\max(\boldsymbol{U}_1)} = \begin{pmatrix} 1 \\ 0.8 \\ 0.1 \end{pmatrix}$.

将 $\boldsymbol{V}_1$ 代入迭代公式(5.20),得 $\boldsymbol{U}_2$ 后,求 $m_2 = \max(\boldsymbol{U}_2)$,再代入(5.20),求 $\boldsymbol{V}_2$,如此下去,将计算结果列入表 5-2.

表 5-2

k	$\lambda_1 \approx m_k = \max(\boldsymbol{U}_k)$	$\boldsymbol{V}_k^{\mathrm{T}} = \boldsymbol{U}_k^{\mathrm{T}}/\max(\boldsymbol{U}_k)$
0		(1,1,1)
1	10	(1,0.8,0.1)
2	7.200 000	(1,0.750 000, -0.111 111)
3	6.500 000	(1,0.730 769, -0.188 034)
4	6.230 769	(1,0.722 222, -0.220 850)
5	6.111 111	(1,0.718 182, -0.235 915)
6	6.054 545	(1,0.716 216, -0.243 095)
7	6.027 027	(1,0.715 247, -0.246 588)
8	6.013 453	(1,0.714 765, -0.248 306)
9	6.006 711	(1,0.714 525, -0.249 157)

续表

k	$\lambda_1 \approx m_k = \max(U_k)$	$V_k^T = U_k^T/\max(U_k)$
10	6.003 352	(1,0.714 405, −0.249 579)
11	6.001 675	(1,0.714 346, −0.249 790)
12	6.000 837	(1,0.714 316, −0.249 895)

例 5.2.1 的计算结果,迭代 12 次,A 的主特征值 λ_1 的近似值 $m_{12} \approx 6.000\,837$和对应的近似特征向量 $V_{12} = (1, 0.714\,316, -0.249\,895)^T$. 其实,$A$ 的特征值为 $\lambda_1 = 6, \lambda_2 = 3, \lambda_3 = 2$.

5.2.2 幂法的 MATLAB 程序

设 n 阶实矩阵 A 的 n 个特征值为 $\lambda_1, \lambda_2, \cdots, \lambda_n$,且满足 $|\lambda_1| > |\lambda_2| \geqslant \cdots \geqslant |\lambda_n| > 0$,$A$ 的主特征值 λ_1 对应的特征向量为 X_1,则我们可以用下面的 MATLAB 程序计算 λ_1 和 X_1 的近似值和近似向量.

用幂法计算矩阵 A 的主特征值和对应的特征向量的 MATLAB 主程序

输入的量:n 阶实矩阵 A、n 维初始实向量 V_0、计算要求的精度 jd、迭代的最大次数 *max*1;

输出的量:迭代的次数 k、A 的主特征值 λ_1 的近似值 *lambda*、λ_1 对应的特征向量 X_1 的近似向量 V_k、相邻两次迭代的误差 Wc. 如果迭代次数已经达到最大的迭代次数 *max*1,则给出提示的相关信息.

根据迭代公式(5.20),现提供用幂法计算矩阵 A 的主特征值和对应的特征向量的 MATLAB 主程序如下:

```
function[k,lambda,Vk,Wc] = mifa(A,V0,jd,max1)
lambda = 0;k = 1;Wc = 1;,jd = jd * 0.1;state = 1;V = V0;
while((k < = max1)&(state = = 1))
Vk = A * V;[mj] = max(abs(Vk));mk = m;
tzw = abs(lambda - mk);Vk = (1/mk) * Vk;
Txw = norm(V - Vk);Wc = max(Txw,tzw);V = Vk;lambda = mk;state = 0;
    if(Wc > jd)
        state = 1;
    end
        k = k + 1;Wc = Wc;
  end
  if(Wc < = jd)
     disp('请注意:迭代次数 k,主特征值的近似值 lambda,主特征向量的近似
```

```
        向量 Vk,相邻两次迭代的误差 Wc 如下:')
else
    disp('请注意:迭代次数 k 已经达到最大迭代次数 max1,主特征值的迭代值
    lambda,主特征向量的迭代向量 Vk,相邻两次迭代的误差 Wc 如下:')
end
  Vk = V;k = k - 1;Wc;
```

例 5.2.2 用幂法计算下列矩阵的主特征值和对应的特征向量的近似向量,精度 $\varepsilon = 10^{-5}$. 并把(1)和(2)输出的结果与例 5.1.1 中的结果进行比较.

(1) $\boldsymbol{A}=\begin{pmatrix}1 & -1\\ 2 & 4\end{pmatrix}$; (2) $\boldsymbol{B}=\begin{pmatrix}1 & 2 & 3\\ 2 & 1 & 3\\ 3 & 3 & 6\end{pmatrix}$;

(3) $\boldsymbol{C}=\begin{pmatrix}1 & 2 & 2\\ 1 & -1 & 1\\ 4 & -12 & 1\end{pmatrix}$; (4) $\boldsymbol{D}=\begin{pmatrix}-4 & 14 & 0\\ -5 & 13 & 0\\ -1 & 0 & 2\end{pmatrix}$.

解 (1) 输入 MATLAB 程序

```
>>A = [1 -1;2  4];V0 = [1,1]';
[k,lambda,Vk,Wc] = mifa(A,V0,0.00001,100),
[V,D] = eig(A),Dzd = max(diag(D)),
wuD = abs(Dzd - lambda),wuV = V(:,2)./Vk,
```

运行后屏幕显示结果

请注意:迭代次数 k,主特征值的近似值 lambda,主特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:

```
k =        lambda =                        Wc =
  33             3.00000173836804              8.691862856124999e-007
Vk =                           V =
  -0.49999942054432    -0.70710678118655     0.44721359549996
   1.00000000000000     0.70710678118655    -0.89442719099992
wuV =                              Dzd =       wuD =
  -0.89442822756294                  3           1.738368038406435e-006
  -0.89442719099992
```

由输出结果可看出,迭代 33 次,相邻两次迭代的误差 $Wc \approx 8.691\,9\mathrm{e}-007$,矩阵 $\boldsymbol{A}$ 的主特征值的近似值 $lambda \approx 3.000\,00$ 和对应的特征向量的近似向量 $\boldsymbol{V}_k \approx (-0.500\,00, 1.000\,00)^{\mathrm{T}}$,$lambda$ 与例 5.1.1 中 $\boldsymbol{A}$ 的最大特征值 $\lambda_2 = 3$ 近似相等,绝对误差约为 $1.738\,37\mathrm{e}-006$,$\boldsymbol{V}_k$ 与特征向量 $\boldsymbol{X}_2^{\mathrm{T}} = k_2\left(\frac{1}{2}, -1\right)^{\mathrm{T}}$ ($k_2 \neq 0$)的第 1 个分量的绝对误差约等于 0,第 2 个分量的绝对值相同. 由 $\boldsymbol{wuV}$ 可以看出,λ_2 的特征向量 $\boldsymbol{V}(:,2)$ 与 $\boldsymbol{V}_k$ 的对应分量的比值近似相等. 因此,用程序

mifa.m计算的结果达到预先给定的精度 $\varepsilon=10^{-5}$.

(2) 输入 MATLAB 程序

```
>>B=[1  2  3;2  1  3;3  3  6];V0=[1,1,1]';
[k,lambda,Vk,Wc]=mifa(B,V0,0.00001,100),[V,D]=eig(B),
Dzd=max(diag(D)),wuD=abs(Dzd-lambda),wuV=V(:,3)./Vk,
```

运行后屏幕显示结果

```
请注意:迭代次数 k,主特征值的近似值 lambda,主特征向量的近似向量 Vk,相邻
两次迭代的误差 Wc 如下:
k =           lambda =          Wc =        Dzd =      wuD =
  3                    9           0            9          0
Vk =                                          wuV =
   0.50000000000000                               0.81649658092773
   0.50000000000000                               0.81649658092773
   1.00000000000000                               0.81649658092773
V =
  0.70710678118655     0.57735026918963   0.40824829046386
 -0.70710678118655     0.57735026918963   0.40824829046386
                  0   -0.57735026918963   0.81649658092773
```

由输出结果可看出,迭代 3 次,相邻两次迭代的误差 $Wc=0$,实对称矩阵 $\boldsymbol{B}$ 的主特征值的近似值 $lambda=9$ 和对应的特征向量的近似向量 $\boldsymbol{V}_k=(0.500\,00,0.500\,00,1.000\,00)^{\mathrm{T}}$,$lambda$ 与例 5.1.1 中 $\boldsymbol{B}$ 的最大特征值 $\lambda_3=9$ 相同,$\boldsymbol{V}_k$ 与特征向量 $\boldsymbol{X}_3^{\mathrm{T}}=k_3(1,1,2)^{\mathrm{T}}(k_3\neq 0)$ 的对应分量成比例. 从 $\boldsymbol{wuV}$ 的每个分量的值也可以看出,λ_3 的特征向量 $\boldsymbol{V}(:,3)$ 与 $\boldsymbol{V}_k$ 的对应分量的比值相等. 因此,用程序 mifa.m 计算的结果达到预先给定的精度 10^{-5}.

此例说明,幂法对实对称矩阵 $\boldsymbol{B}$ 的迭代速度快且计算结果精度高,

(3) 输入 MATLAB 程序

```
>>C=[1  2  2;1 -1  1;4 -12  1];V0=[1,1,1]';
[k,lambda,Vk,Wc]=mifa(C,V0,0.00001,100),[V,D]=eig(C),
Dzd = max(diag(D)),wuD = abs(Dzd - lambda),Vzd = V(:,1),wuV =
V(:,1)./Vk,
```

运行后屏幕显示

```
请注意:迭代次数 k 已经达到最大迭代次数 max1,主特征值的迭代值 lambda,主
特征向量的迭代向量 Vk,相邻两次迭代的误差 Wc 如下:
k =            lambda =                     Wc =
 100                 0.09090909090910     2.37758124193119
Dzd =                                     wuD =
    1.00000000000001                          0.90909090909091
```

```
Vk =                Vzd =               wuV =
0.99999999999993    0.90453403373329    0.90453403373335
0.99999999999995    0.30151134457776    0.30151134457778
1.00000000000000   -0.30151134457776   -0.30151134457776
```

由输出结果可见,迭代次数 k 已经达到最大迭代次数 $max1=100$,并且 *lambda* 的相邻两次迭代的误差 $Wc\approx 2.377\ 58>2$,由 ***wuV*** 可以看出,*lambda* 的特征向量 V_k 与真值 *Dzd* 的特征向量 ***Vzd*** 对应分量的比值相差较大,所以迭代序列发散. 实际上,实数矩阵 $\boldsymbol{C}$ 的特征值的近似值为 $\lambda_1=1.00000000000001$,$\lambda_2=-\mathrm{i}$,$\lambda_3=\mathrm{i}$,并且对应的特征值向量的近似向量分别为 $\boldsymbol{X}_1^{\mathrm{T}}=k_1(0.90453403373329,$ $0.30151134457776,-0.30151134457776)^{\mathrm{T}}$,$\boldsymbol{X}_2^{\mathrm{T}}=k_2(-0.72547625011001,$ $-0.21764287503300-0.07254762501100\mathrm{i},0.58038100008801\ -0.29019050004400\mathrm{i})^{\mathrm{T}}$, $\boldsymbol{X}_3^{\mathrm{T}}=k_3(-0.72547625011001,-0.21764287503300+0.07254762501100\mathrm{i},$ $0.58038100008801+0.29019050004400\mathrm{i})^{\mathrm{T}}$,($k_1\neq 0,k_2\neq 0,k_3\neq 0$ 是常数).

此例说明,当 n **阶实矩阵有复数特征值时,不宜用幂法计算它的主特征值** λ_1 **对应的特征向量** $\boldsymbol{X}_1$**.**

(4) 输入 MATLAB 程序

```
>>D=[-4  14  0;-5  13  0;-1  0  2];V0=[1,1,1]';
[k,lambda,Vk,Wc]=mifa(D,V0,0.00001,100),[V,Dt]=eig(D),
Dtzd=max(diag(Dt)),wuDt=abs(Dtzd-lambda),Vzd=V(:,2),wuV=
V(:,2)./Vk,
```

运行后屏幕显示结果

请注意:迭代次数 k,主特征值的近似值 lambda,主特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:

```
k =    lambda =                  Wc =
 19          6.00000653949528      6.539523793591684e-006
Dtzd =                        wuDt =
      6.00000000000000          6.539495284840768e-006
Vk =                 Vzd =                wuV =
  0.79740048053564     0.79740048053564    0.79740048053564
  0.71428594783886     0.56957177181117    0.79740021980618
 -0.24999918247180    -0.19935012013391    0.79740308813370
```

由输出结果可见,迭代 19 次,相邻两次迭代的误差 $Wc\approx 6.539\ 52\mathrm{e}-006$,矩阵 $\boldsymbol{D}$ 的主特征值的近似值 $lambda\approx 6.000\ 01$ 和对应的特征向量的近似向量为 $\boldsymbol{V}_k\approx(0.797\ 40,0.714\ 29,-0.250\ 00)^{\mathrm{T}}$. 用 eig(A)计算,*lambda* 与对应的特征值的真值 *Dtzd* 的绝对误差 $wuDt<10^{-5}$,二者的特征向量 $\boldsymbol{V}(:,2)$ 与 ***Vk*** 的对应分量的比值近似相等(请对比 ***wuV*** 的每个分量的值). 因此,用程序 mifa.m 计

算的结果达到预先给定的精度 $\varepsilon=10^{-5}$.

由例 5.2.2 的计算结果可见,用幂法计算实对称矩阵 B 的主特征值对应的特征向量时,得到的迭代序列的收敛速度最快且计算结果精度也最高;非实对称矩阵对应的迭代序列的敛散性不定,有时发散(例如矩阵 C),有时收敛(例如矩阵 A 和 D),且收敛速度较慢,比较(1)和(4)的计算结果可知,用幂法得到的迭代序列的收敛速度与矩阵的阶数无关;当实矩阵有复数特征值时,不宜用幂法计算它的主特征值对应的特征向量.



习 题 5.2

1. 用手工计算 $A=\begin{pmatrix}0 & 11 & -5\\ -2 & 17 & -7\\ -4 & 26 & -10\end{pmatrix}$ 的主特征值和对应的近似特征向量,迭代 4 次.

2. 用幂法计算下列矩阵的主特征值和对应的近似特征向量,精度为 0.000 01. 将计算结果与习题 5.1 第 1 题中的(1)和(2)的计算结果比较.

(1) $A=\begin{pmatrix}-5 & 2 & 17\\ 2 & -8 & 3\\ 3 & 3 & -3\end{pmatrix}$;(2) $B=\begin{pmatrix}-11 & 2 & 15\\ 2 & 58 & 3\\ 15 & 3 & -3\end{pmatrix}$.

5.3 反幂法和位移反幂法及其 MATLAB 程序

反幂法是求非奇异矩阵 A 的按模最小特征值及其对应的特征向量的一种迭代方法. 本节主要介绍反幂法和原点位移反幂法及其 MATLAB 程序.

5.3.1 反幂法

设 A 为 n 阶非奇异矩阵,则 A 的特征值 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 都不全为零,且满足

$$|\lambda_1|\geqslant|\lambda_2|\geqslant\cdots\geqslant|\lambda_n|>0.$$

从而有

$$AX_i=\lambda_i X_i\ \ (i=1,2,\cdots,n). \tag{5.21}$$

因为 A^{-1}存在,如果 $AX_i=\lambda_i X_i$,则有 $A^{-1}X_i=\lambda_i^{-1}X_i$. 因而,$A^{-1}$的特征值为 λ_1^{-1},$\lambda_2^{-1},\cdots,\lambda_n^{-1}$,其次序为

$$|\lambda_1^{-1}|\leqslant|\lambda_2^{-1}|\leqslant\cdots\leqslant|\lambda_{n-1}^{-1}|\leqslant|\lambda_n^{-1}|.$$

这样,计算 A 的按模最小特征值 λ_n,就变成对 A^{-1}应用幂法,即称为对 A 应用**反幂法**,求 A^{-1}的主特征值 λ_n^{-1}.

我们讨论 λ_n 是实数而且是单重特征值的情形，此时有

$$|\lambda_1^{-1}| \leqslant |\lambda_2^{-1}| \leqslant \cdots \leqslant |\lambda_{n-1}^{-1}| < |\lambda_n^{-1}|. \tag{5.22}$$

反幂法的基本思想是任取一个非零 n 维初始实向量 $\boldsymbol{V}_0 \neq \boldsymbol{O}$，用迭代公式

$$\begin{cases} \boldsymbol{U}_k = \boldsymbol{A}^{-1}\boldsymbol{V}_{k-1}, \\ \boldsymbol{V}_k = \dfrac{\boldsymbol{U}_k}{\max(\boldsymbol{U}_k)} \end{cases} \quad (k=1,2,\cdots), \tag{5.23}$$

其中 $\boldsymbol{U}_k = \{u_k^{(1)}, u_k^{(2)}, \cdots, u_k^{(n)}\}$，$\max(\boldsymbol{U}_k) = \max\limits_{1\leqslant i\leqslant n}\{|u_k^{(i)}|\}$ $(k=1,2,\cdots,n)$.

由(5.22)、(5.17)和(5.19)式知，

$$\lim_{k\to\infty}\max(\boldsymbol{U}_k) = \frac{1}{\lambda_n}, \tag{5.24}$$

且

$$\lim_{k\to\infty}\boldsymbol{V}_k = \frac{\boldsymbol{X}_n}{\max(\boldsymbol{X}_n)}. \tag{5.25}$$

因此，当 k 充分大时，$\lambda_n \approx \dfrac{1}{\max(\boldsymbol{U}_k)}$，且 $\boldsymbol{V}_k \approx \dfrac{\boldsymbol{X}_n}{\max(\boldsymbol{X}_n)}$.

定理 5.18 如果 n 阶非奇异矩阵 $\boldsymbol{A}$ 有 n 个线性无关的特征向量 $\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n$，对应的特征值 $\lambda_1, \lambda_2, \cdots, \lambda_n$ 满足 $|\lambda_1| \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_{n-1}| > |\lambda_n|$，则对于任何非零 n 维初始向量 $\boldsymbol{V}_0 = \boldsymbol{U}_0$，由反幂法的迭代公式(5.23)构造的迭代序列 $\{\boldsymbol{V}_k\}$ 和 $\{\max(\boldsymbol{U}_k)\}$ 满足

$$\lim_{k\to\infty}\boldsymbol{V}_k = \frac{\boldsymbol{X}_n}{\max(\boldsymbol{X}_n)} \quad 且 \quad \lim_{k\to\infty}\max(\boldsymbol{U}_k) = \lambda_n^{-1}.$$

实际计算时，求 $\boldsymbol{A}^{-1}$ 不方便，所以将(5.23)式中的 $\boldsymbol{U}_k = \boldsymbol{A}^{-1}\boldsymbol{V}_{k-1}$ 改写成 $\boldsymbol{A}\boldsymbol{U}_k = \boldsymbol{V}_{k-1}$，并将 $\boldsymbol{A}$ 作三角分解，即 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$，其中 $\boldsymbol{L}$ 为单位下三角形矩阵，$\boldsymbol{U}$ 为上三角形矩阵. 如果任取一个非零 n 维初始实向量 $\boldsymbol{V}_0 \neq \boldsymbol{O}$，则反幂法的迭代公式(5.23)可改写成

$$\begin{cases} \boldsymbol{L}\boldsymbol{Y}_k = \boldsymbol{V}_{k-1}, \\ \boldsymbol{Y}_k = \boldsymbol{U}\boldsymbol{U}_k, \\ \boldsymbol{V}_k = \dfrac{\boldsymbol{U}_k}{\max(\boldsymbol{U}_k)} \end{cases} \quad (k=1,2,\cdots). \tag{5.26}$$

5.3.2 原点位移反幂法

当 $\left|\dfrac{\lambda_n}{\lambda_{n-1}}\right| \approx 1$ 时，由(5.26)所生成的迭代序列收敛速度较慢，为加快速度，在实际计算中总是采用带原点位移的反幂法，即以 $\boldsymbol{A} - a\boldsymbol{E}$ 代替 $\boldsymbol{A}$，使得关于 $\boldsymbol{A} - a\boldsymbol{E}$ 的反幂法具有较快的收敛速度. 这种方法称为**原点位移反幂法**. 这里的位

移值通常取 λ_n 的某一近似值 $\tilde{\lambda}_n$.

定理 5.19 如果 λ_i 是 n 阶非奇异矩阵 $\boldsymbol{A}$ 的特征值,$\boldsymbol{X}_i$ 是 A 的对应于 λ_i 的特征向量,如果 a 是任意常量,则 λ_i-a 是矩阵 $\boldsymbol{A}-a\boldsymbol{E}$ 的特征值,$\boldsymbol{X}_i$ 是 $\boldsymbol{A}-a\boldsymbol{E}$ 的对应于 λ_i-a 的特征向量.

定理 5.20 如果 λ_i 是 n 阶非奇异矩阵 $\boldsymbol{A}$ 的特征值,$\boldsymbol{X}_i$ 是 $\boldsymbol{A}$ 的对应于 λ_i 的特征向量,且任意常量 $a\neq\lambda_i$,则 $1/(\lambda_i-a)$ 是矩阵 $(\boldsymbol{A}-a\boldsymbol{E})^{-1}$ 的特征值,$\boldsymbol{X}_i$ 是 $(\boldsymbol{A}-a\boldsymbol{E})^{-1}$ 的对应于 $1/(\lambda_i-a)$ 的特征向量.

定理 5.21 设 n 阶非奇异矩阵 $\boldsymbol{A}$ 有不同的特征值 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 且都不全为零,选择常数 $\tilde{\lambda}_n\neq\lambda_n$,使得 $\sigma_1=(\lambda_n-\tilde{\lambda}_n)^{-1}$ 满足

$$|(\lambda_1-\tilde{\lambda}_n)^{-1}|\leqslant|(\lambda_2-\tilde{\lambda}_n)^{-1}|\leqslant\cdots\leqslant|(\lambda_{n-1}-\tilde{\lambda}_n)^{-1}|<|(\lambda_n-\tilde{\lambda}_n)^{-1}|.$$

如果选择适当的 n 维初始实向量 $\boldsymbol{U}_0\neq\boldsymbol{O}$,则由原点位移反幂法的迭代公式

$$\begin{cases}\boldsymbol{V}_k=(\boldsymbol{A}-\tilde{\lambda}_n\boldsymbol{E})^{-1}\boldsymbol{U}_{k-1},\\ \boldsymbol{U}_k=\dfrac{\boldsymbol{V}_k}{\max(\boldsymbol{V}_k)}\end{cases}\quad(k=1,2,\cdots),\tag{5.27}$$

其中 $\max(\boldsymbol{V}_k)=\max\limits_{1\leqslant i\leqslant n}\{|v_k^{(i)}|\},\boldsymbol{V}_k=\{v_k^{(1)},v_k^{(2)},\cdots,v_k^{(n)}\}\ (k=1,2,\cdots,n)$ 所生成的迭代序列 $\{\max(\boldsymbol{V}_k)\}$ 和 $\{\boldsymbol{U}_k\}$ 分别收敛到矩阵 $(\boldsymbol{A}-\tilde{\lambda}_n\boldsymbol{E})^{-1}$ 的主特征值 $(\lambda_n-\tilde{\lambda}_n)^{-1}$ 和主特征向量 $\boldsymbol{X}_n$. A 的特征值 $\lambda_n=\dfrac{1}{\sigma_1}+\tilde{\lambda}_n$.

注意 选择一个数 $\tilde{\lambda}_n(\tilde{\lambda}_n\neq\lambda_i)$,使得它最接近 λ_n,如果 $|\lambda_n-\tilde{\lambda}_n|<|\lambda_j-\tilde{\lambda}_n|\ (j<n)$,则收敛速度由比值 $\dfrac{|\lambda_n-\tilde{\lambda}_n|}{\min|\lambda_j-\tilde{\lambda}_n|}\ (j<n)$ 确定,$\tilde{\lambda}_n$ 越精确,收敛的速度越快.

带原点位移的反幂法的计算量大大地超过了幂法,因为每步计算 $\boldsymbol{V}_k$ 时,需要解线性方程组,好在这些线性方程组的系数矩阵 $\boldsymbol{A}-\tilde{\lambda}_n\boldsymbol{E}$ 相同,可预先将 $\boldsymbol{A}-\tilde{\lambda}_n\boldsymbol{E}$ 作三角分解,即 $\boldsymbol{A}-\tilde{\lambda}_n\boldsymbol{E}=\tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$,其中 $\tilde{\boldsymbol{L}}$ 为单位下三角形矩阵,$\tilde{\boldsymbol{U}}$ 为上三角形矩阵. 如果任取一个非零 n 维初始实向量 $\boldsymbol{U}_0=\boldsymbol{V}_0\neq\boldsymbol{O}$,则原点位移反幂法的迭代公式(5.27)可改写成

$$\begin{cases}\tilde{\boldsymbol{L}}\boldsymbol{Y}_k=\boldsymbol{U}_{k-1},\\ \boldsymbol{Y}_k=\tilde{\boldsymbol{U}}\boldsymbol{V}_k,\\ \boldsymbol{U}_k=\dfrac{\boldsymbol{V}_k}{\max(\boldsymbol{V}_k)}\end{cases}\quad(k=1,2,\cdots).\tag{5.28}$$

例 5.3.1 用原点位移反幂法的迭代公式(5.28),计算矩阵 $\boldsymbol{A}=\begin{pmatrix}-4&14&0\\-5&13&0\\-1&0&2\end{pmatrix}$ 的与特征值 $\lambda_3=3-\sqrt{3}\approx\tilde{\lambda}_3=1.2679$ 对应的特征向量 $\boldsymbol{X}_3$.

解 用原点位移反幂法,首先三角分解矩阵

$$A-\tilde{\lambda}_n E = A-1.267\,9E=\tilde{L}\tilde{U},$$

其中

$$\tilde{L}=\begin{pmatrix}1.0 & 0 & 0\\ 1.365\,9 & 1.0 & 0.0\\ 0.0 & 2.731\,0 & 1.0\end{pmatrix},\quad \tilde{U}=\begin{pmatrix}0.732\,1 & 1.0 & 0.0\\ 0.0 & 0.366\,2 & 1.0\\ 0.0 & 0.0 & 0.001\,1\end{pmatrix}.$$

取 $U_0=(1,1,1)^{\mathrm{T}}$,根据迭代公式(5.28),即

$$\begin{cases}\tilde{L}Y_k=U_{k-1},\\ Y_k=\tilde{U}V_k,\\ U_k=\dfrac{V_k}{\max(V_k)}\end{cases}\quad (k=1,2,\cdots).$$

解方程组 $Y_1=\tilde{L}^{-1}U_0$ 和 $V_1=\tilde{U}^{-1}Y_1$,得 $V_1=(3\,385.2,-2\,477.3,908.20)^{\mathrm{T}}$,

$$U_1=\frac{V_1}{\max(V_1)}=(1.000\,0,-0.731\,8,0.268\,3)^{\mathrm{T}}.$$

再解方程组 $Y_2=\tilde{L}^{-1}U_1$ 和 $V_2=\tilde{U}^{-1}Y_2$,得 $V_2=(20\,345,-14\,894,5\,451.9)^{\mathrm{T}}$,

$$U_2=\frac{V_2}{\max(V_2)}=(1.000\,0,-0.732\,1,0.269\,8)^{\mathrm{T}}.$$

再解方程组 $Y_3=\tilde{L}^{-1}U_2$ 和 $V_3=\tilde{U}^{-1}Y_3$,得 $V_3=V_2=(20\,345,-14\,894,5\,451.9)^{\mathrm{T}}$. 实际上,与 λ_3 对应的精确特征向量

$$X_3=(1,1-\sqrt{3},2-\sqrt{3})^{\mathrm{T}}=(1.000\,0,-0.732\,1,0.268\,0)^{\mathrm{T}}.$$

5.3.3 原点位移反幂法的两种 MATLAB 程序

设 n 阶实矩阵 A 的 n 个特征值 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 都不相同,且满足 $|\lambda_n-\tilde{\lambda}_n|<|\lambda_j-\tilde{\lambda}_n|\ (j<n)$,且 A 的按模最小特征值 λ_n 对应的特征向量为 X_n,则我们可以根据迭代公式(5.27)和(5.28)编写两种 MATLAB 程序分别计算 λ_n 和 X_n 的近似值和近似向量.

(一)原点位移反幂法的 MATLAB 主程序 1

根据原点位移反幂法的迭代公式(5.28),现提供用原点位移反幂法计算矩阵 A 的按模最小特征值和对应的特征向量的 MATLAB 主程序如下:

用原点位移反幂法计算矩阵 A 的特征值和对应的特征向量的 MATLAB 主程序 1

输入的量:n 阶实矩阵 A、n 维初始实向量 V_0、特征值的近似值 *jlamb*、计算的精度 *jd*、迭代的最大次数 *max*1;

输出的量:迭代的次数 k、A 的特征值 λ_n 的近似值 $lambda_n$、与 λ_n 对应的特征向量 X_n 的近似向量 V_k、相邻两次迭代的误差 *Wc*. 如果迭代次数已经达到最大的迭代次数 *max*1,则给出提示的相关信息.

```
function[k,lambdan,Vk,Wc] = ydwyfmf(A,V0,jlamb,jd,max1)
[n,n] = size(A);A1 = A - jlamb * eye(n);jd = jd * 0.1;RA1 = det(A1);
if RA1 = = 0
disp('请注意:因为 A - aE 的 n 阶行列式 h1 等于零,所以 A - aE 不能进行 LU 分解.')
return
end
lambda = 0;
if RA1 ~ = 0
   for p = 1:n
      h(p) = det(A1(1:p,1:p));
   end
h1 = h(1:n);
fori = 1:n
  if  h(1,i) = = 0
     disp('请注意:因为 A - aE 的 r 阶主子式等于零,所以 A - aE 不能进行 LU 分解.')
     return
   end
  end
if  h(1,i) ~ = 0
     disp('请注意:因为 A - aE 的各阶主子式都不等于零,所以 A - aE 能进行 LU 分解.')
     k = 1;Wc = 1;state = 1;Vk = V0;
   while ((k < = max1)&(state = = 1))
        [L U] = lu(A1);Yk = L \Vk;Vk = U \Yk;[mj] = max(abs(Vk));
        mk = m;Vk1 = Vk /mk;Yk1 = L \Vk1;Vk1 = U \Yk1;
        [m j] = max(abs(Vk1));
        mk1 = m;Vk2 = (1 /mk1) * Vk1;tzw1 = abs((mk - mk1) /mk1);
        tzw2 = abs(mk1 - mk);Txw1 = norm(Vk) - norm(Vk1);
        Txw2 = (norm(Vk) - norm(Vk1)) /norm(Vk1);
        Txw = min(Txw1,Txw2);tzw = min(tzw1,tzw2);Vk = Vk2;
        mk = mk1;Wc = max(Txw,tzw);Vk = Vk2;mk = mk1;state = 0;
     if(Wc > jd)
          state = 1;
     end
        k = k + 1;
   end
```

```
        if(Wc < =jd)
            disp('A - aE 的秩 R(A - aE)和各阶顺序主子式值 hl、迭代次数 k,按模最小特征值的近似值 lambda,特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:')
        else
            disp('A - aE 的秩 R(A - aE)和各阶顺序主子式值 hl、迭代次数 k 已经达到最大迭代次数 max1,按模最小特征值的迭代值 lambda,特征向量的迭代向量 Vk,相邻两次迭代的误差 Wc 如下:')
        end
         hl,RA1
        end
    end
    [V,D] =eig(A,'nobalance'),Vk;k =k -1;Wc;lambdan =jlamb +1 /mk1;
```

例 5.3.2 用原点位移反幂法的迭代公式(5.28),根据给定的下列矩阵的特征值 λ_n 的初始值 $\tilde{\lambda}_n$,计算与 λ_n 对应的特征向量 $\boldsymbol{X}_n$ 的近似向量,精确到0.000 1.

(1) $\begin{pmatrix} 1 & -1 & 0 \\ -2 & 4 & -2 \\ 0 & -1 & 2 \end{pmatrix}$, $\tilde{\lambda}_2=0.2$; (2) $\begin{pmatrix} 1 & -1 \\ 2 & 4 \end{pmatrix}$, $\tilde{\lambda}_2=2.001$;

(3) $\begin{pmatrix} -11 & 2 & 15 \\ 2 & 58 & 3 \\ 15 & 3 & -3 \end{pmatrix}$, $\tilde{\lambda}_3=8.26$.

解 (1) 输入 MATLAB 程序

```
>>A =[1 -1 0; -2  4 -2;0 -1 2];V0 =[1,1,1]';
[k,lambda,Vk,Wc] =ydwyfmf(A,V0,0.2,0.0001,10000)
```

运行后屏幕显示结果

```
请注意:因为 A - aE 的各阶主子式都不等于零,所以 A - aE 能进行 LU 分解.
A - aE 的秩 R(A - aE)和各阶顺序主子式值 hl、迭代次数 k,按模最小特征值的近似值 lambda,特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:
k =        lambda =       Wc =               hl =
 3              0.2384     1.0213e-007      0.8000   1.0400   0.2720
Vk =       V =                               D =
    1.0000   -0.2424   -1.0000   -0.5707   5.1249        0         0
    0.7616    1.0000   -0.7616    0.3633        0   0.2384         0
    0.4323   -0.3200   -0.4323    1.0000        0        0    1.6367
```

(2) 输入 MATLAB 程序

```
>>A =[1 -1;2  4];V0 =[20,1]';
[k,lambda,Vk,Wc] =ydwyfmf(A,V0,2.001,0.0001,100)
```

运行后屏幕显示结果

请注意:因为 A - aE 的各阶主子式都不等于零,所以 A - aE 能进行 LU 分解.

A - aE 的秩 R(A - aE)和各阶顺序主子式值 hl、迭代次数 k,按模最小特征值的近似值 lambda,特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:

```
k =       lambda =      Wc =                hl =
  2          2.0020        5.1528e-007        -1.0010     -0.0010
Vk =          V =                          D =
  1.0000           -1.0000     0.5000          2   0
 -1.0000            1.0000    -1.0000          0   3
```

(3) 输入 MATLAB 程序

```
>>A =[-11  2  15;2  58  3;15  3 -3];V0 =[1,1,-1]';
[k,lambdan,Vk,Wc] =ydwyfmf(A,V0,8.26,0.0001,100)
```

运行后屏幕显示结果

请注意:因为 A - aE 的各阶主子式都不等于零,所以 A - aE 能进行 LU 分解.

A - aE 的秩 R(A - aE)和各阶顺序主子式值 hl、迭代次数 k,按模最小特征值的近似值 lambda,特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:

```
k =   lambdan =     Wc =               hl =
  2        8.2640      6.9304e-008       -19.2600 -961.9924    -6.1256
Vk =        V =                            D =
   -0.7692      0.7928      0.6081   0.0416    -22.5249          0          0
    0.0912      0.0030      0.0721   0.9974           0     8.2640          0
   -1.0000     -0.6095      0.7906   0.0590           0          0    58.2609
```

例 5.3.3 用原点位移反幂法的迭代公式(5.28),计算 $A=\begin{pmatrix}0 & 11 & -5\\ -2 & 17 & -7\\ -4 & 26 & -10\end{pmatrix}$的分别对应于特征值 $\lambda_1\approx\tilde{\lambda}_1=1.001$, $\lambda_2\approx\tilde{\lambda}_2=2.001$, $\lambda_3\approx\tilde{\lambda}_3=4.001$ 的特征向量 X_1, X_2, X_3 的近似向量,相邻迭代误差为 0.001. 将计算结果与精确特征向量比较,其中

$\lambda_1=0.999\,999\,999\,999\,97\cdots$, $\lambda_2=1.999\,999\,999\,999\,99\cdots$, $\lambda_3=4.000\,000\,000\,000\,02\cdots$,

分别对应特征向量为

$X_1=(0.500\,000\,000\,000\,00, 0.500\,000\,000\,000\,00, 1.000\,000\,000\,000\,00,)^{T}$

$X_2=(-0.249\,999\,999\,999\,99, -0.500\,000\,000\,000\,00, -1.000\,000\,000\,000\,00)^{T}$,

$X_3=(-0.400\,000\,000\,000\,00, -0.600\,000\,000\,000\,00, -1.000\,000\,000\,000\,00)^{T}$.

解 (1) 计算特征值 $\lambda_1\approx\tilde{\lambda}_1=1.001$ 对应的特征向量 X_1 的近似向量. 输入 MATLAB 程序

```
>>A =[0  11 -5;-2  17 -7;-4  26 -10];V0 =[1,1,1]';
[k,lambda,Vk,Wc] =ydwyfmf(A,V0,1.001,0.001,100),[V,D] =eig(A);
Dzd =min(diag(D)),wuD =abs(Dzd -lambda),VD =V(:,1),wuV =V(:,1)./Vk,
```

运行后屏幕显示结果

```
请注意:因为 A - aE 的各阶主子式都不等于零,所以 A - aE 能进行 LU 分解.
A - aE 的秩 R(A - aE)和各阶顺序主子式值 hl、迭代次数 k,按模最小特征值的近似值 lambda,特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:
hl =
  -1.00100000000000    5.98500100000000    -0.00299600100000
k =        lambda =                    RA1 =
  5             1.00200000000000         -0.00299600100000
Vk =                      VD =                     wuV -
  -0.50000000000000     -0.40824829046386     0.81649658092773
  -0.50000000000000     -0.40824829046386     0.81649658092773
  -1.00000000000000     -0.81649658092773     0.81649658092773
Wc =                       Dzd =                     wuD =
  1.378794763695562e-009  1.00000000000000   0.00200000000000
```

从输出的结果可见,迭代 5 次,特征向量 X_1 的近似向量 $\tilde{X}_1$ 的相邻两次迭代的误差 $Wc \approx 1.379e-009$,由 wuV 可以看出,$\tilde{X}_1 = V_k$ 与 VD 的对应分量的比值相等. 特征值 λ_1 的近似值 $lambda \approx 1.002$ 与初始值 $\tilde{\lambda}_1 = 1.001$ 的绝对误差为 0.001,而与 λ_1 的绝对误差为 0.002,其中

$X_1 = (-0.500\,000\,000\,000\,00\cdots, -0.500\,000\,000\,000\,00\cdots, 1.000\,000\,000\,000\,00\cdots)^T$,

$\tilde{X}_1 = (-0.500\,000\,000\,000\,00, -0.500\,000\,000\,000\,00, 1.000\,000\,000\,000\,00)^T$.

(2) 计算特征值 $\lambda_2 \approx \tilde{\lambda}_2 = 2.001$ 对应特征向量 X_2 的近似向量.

输入 MATLAB 程序

```
>>A=[0  11 -5; -2  17 -7; -4  26 -10];V0=[1,1,1]';
[k,lambda,Vk,Wc]=ydwyfmf(A,V0,2.001,0.001,100),
[V,D]=eig(A);WD=lambda-D(2,2),VD=V(:,2),wuV=V(:,2)./Vk,
```

运行后屏幕显示结果

```
请注意:因为 A - aE 的各阶主子式都不等于零,所以 A - aE 能进行 LU 分解.
A - aE 的秩 R(A - aE)和各阶顺序主子式值 hl、迭代次数 k,按模最小特征值的近似值 lambda,特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:
hl =
  -2.00100000000000    -8.01299900000000    0.00200099900000
k =  Wc =                        lambda =             WD =
2  3.131363162302120e-007   2.00200000000016    0.00200000000016
Vk =                     VD =                  wuV =
-0.24999999999999  0.21821789023599  -0.87287156094401
-0.49999999999999  0.43643578047198  -0.87287156094398
-1.00000000000000  0.87287156094397  -0.87287156094397
```

从输出的结果可见,迭代 2 次,特征向量 X_2 的近似向量 $\tilde{X}_2$ 的相邻两次迭代的误差 $Wc \approx 3.131e-007$,$\tilde{X}_2$ 与 X_2 的对应分量的比值近似相等. 特征值 λ_2 的近似值 $lambda \approx 2.002$ 与初始值 $\tilde{\lambda}_2 = 2.001$ 的绝对误差约为 0.001,而 $lambda$ 与 λ_2 的绝对误差约为 0.002,其中

$\tilde{X}_2 = (-0.249\ 999\ 999\ 999\ 99, -0.499\ 999\ 999\ 999\ 99, -1.000\ 000\ 000\ 000\ 00)^{T}$,

$X_2 = (-0.24999999999999, -0.50000000000000, -1.00000000000000\cdots)^{T}$.

(3) 计算特征值 $\lambda_3 \approx \tilde{\lambda}_3 = 4.001$ 对应特征向量 X_3 的近似向量. 输入 MATLAB 程序

```
>>A=[0  11 -5; -2  17 -7; -4  26 -10];V0=[1,1,1]';
[k,lambda,Vk,Wc]=ydwyfmf(A,V0,4.001,0.001,100)
[V,D]=eig(A);WD=lambda-max(diag(D)),VD=V(:,3),wuV=V(:,3)./Vk,
```

运行后屏幕显示结果

```
请注意:因为 A-aE 的各阶主子式都不等于零,所以 A-aE 能进行 LU 分解.
A-aE 的秩 R(A-aE)和各阶顺序主子式值 hl、迭代次数 k,按模最小特征值的近似值 lambda,特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:
hl =
     -4.00100000000000   -30.00899900000000   -0.00600500099999
k =      lambda =                         Wc =
  2              4.00199999999990        1.996084182914842e-007
WD =
   0.00199999999990
Vk =                      VD =                      wuV =
    0.40000000000001    -0.32444284226153    -0.81110710565380
    0.60000000000001    -0.48666426339229    -0.81110710565381
    1.00000000000000    -0.81110710565381    -0.81110710565381
```

从输出的结果可见,迭代 2 次,特征向量 X_3 的近似向量 $\tilde{X}_3$ 的相邻两次迭代的误差 $Wc \approx 1.996e-007$,$\tilde{X}_3$ 与 X_3 的对应分量的比值近似相等. 特征值 λ_3 的近似值 $lambda \approx 4.002$ 与初始值 $\tilde{\lambda}_2 = 4.001$ 的绝对误差近似为 0.001,而 $lambda$ 与 λ_3 的绝对误差约为 0.002,其中

$X_3 = (-0.400\ 000\ 000\ 000\ 00, -0.600\ 000\ 000\ 000\ 00, -1.000\ 000\ 000\ 000\ 00)^{T}$,

$\tilde{X}_3 = (0.400\ 000\ 000\ 000\ 1, 0.600\ 000\ 000\ 000\ 01, 1.000\ 000\ 000\ 000\ 000)^{T}$.

综上所述,用原点位移反幂法的迭代公式(5.28)求矩阵的全部特征值分别对应特征向量的收敛速度快,且精确度高. 但是求矩阵的全部特征值的结果不理想.

(二) 原点位移反幂法的 MATLAB 主程序 2

根据迭代公式(5.27),现提供用原点位移反幂法计算矩阵 A 的按模最小特征值和对应的特征向量的 MATLAB 主程序如下:

用原点位移反幂法计算矩阵 A 的特征值和对应的特征向量的 MATLAB 主程序 2

输入的量:n 阶实矩阵 $\boldsymbol{A}$、n 维初始实向量 $\boldsymbol{V}_0$、特征值的近似值 *jlambn*、计算的精度 *jd*、迭代的最大次数 *max*1.

输出的量:迭代的次数 k、$\boldsymbol{A}$ 的特征值 λ_n 的近似值 $lambda_n$、λ_n 对应的特征向量 $\boldsymbol{X}_n$ 的近似向量 $\boldsymbol{V}_k$、相邻两次迭代的误差 Wc. 如果迭代次数已经达到最大的迭代次数 *max*1,则给出提示的相关信息.

```
function[k,lambdan,Vk,Wc] = wfmifa1(A,V0,jlamb,jd,max1)
[n,n] = size(A);jd = jd * 0.1;A1 = A - jlamb * eye(n);nA1 = inv(A1);
lambda1 = 0;k = 1;Wc = 1;state = 1;U = V0;
while((k < = max1)&(state = = 1))
    Vk = A1 \U;[m j] = max(abs(Vk));mk = m;Vk = (1/mk) * Vk;Vk1 = A1 \Vk;
    [m1 j] = max(abs(Vk1));mk1 = m1,Vk1 = (1/mk1) * Vk1;U = Vk1,
    Txw = (norm(Vk1)  norm(Vk))/norm(Vk1);
    tzw = abs((lambda1 - mk1)/mk1);
    Wc = max(Txw,tzw);lambda1 = mk1;state = 0;
if(Wc > jd)
  state = 1;
end
  k = k + 1;
end
if(Wc < = jd)
    disp('请注意迭代次数 k,特征值的近似值 lambda,对应的特征向量的近似
向量 Vk,相邻两次迭代的误差 Wc 如下:')
else
    disp('请注意迭代次数 k 已经达到最大迭代次数 max1,特征值的近似值
lambda,对应的特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:')
end
[V,D] = eig(A,'nobalance'),Vk = U;k = k - 1;Wc;lambdan = jlamb + 1/mk;
```

例 5.3.4 用原点位移反幂法的迭代公式(5.27),计算例题 5.3.3,并且将这两个例题的计算结果进行比较. 再用两种原点位移反幂法的 MATLAB 主程序,求 $\tilde{\lambda}_1 = 0.999\,999\,999\,999\,97$ 对应的特征向量.

解 (1) 计算特征值 $\lambda_1 \approx \tilde{\lambda}_1 = 1.001$ 对应特征向量 X_1 的近似向量.

输入 MATLAB 程序

```
>>A = [0  11 -5; -2  17 -7; -4  26 -10];V0 = [1,1,1]';
[k,lambda,Vk,Wc] = wfmifa1(A,V0,1.001,0.001,100)
```

运行后屏幕显示结果

请注意迭代次数 k,特征值的近似值 lambda,对应的特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:

```
k =    lambda =              Wc =
  5          1.00200000000138  1.376344154436924e-006
Vk' =  -0.50000000000000  -0.50000000000000  -1.00000000000000
```

同理可得,另外与两个特征值对应的特征向量,将计算结果列表 5-3.

表 5-3 例题 5.3.3 和例题 5.3.4 计算结果比较

	由(5.27)式计算结果	由(5.28)式计算结果
迭代次数 k X_1 和其近似向量 V_k $\tilde{X}_1$ 的相邻迭代误差 Wc λ_1 和其近似值 $lambda$	$k=5$ $X_1=(0.500,0.500,1.000)^T$ $V_k=(-0.500,-0.500,-1.000)^T$ $Wc=1.376\ 344\ 154\ 436\ 924e-006$ $lambda=1.002\ 000\ 000\ 001\ 38$ $\lambda_1=0.999\ 999\ 999\ 999\ 97\cdots$	$k=5$ $X_1=(0.500,0.500,1.000)^T$ $V_k=(-0.500,-0.500,-1.000)^T$ $Wc=1.378\ 794\ 763\ 695\ 562e-009$ $lambda=1.002\ 000\ 000\ 000\ 00$ $\lambda_1=0.999\ 999\ 999\ 999\ 97\cdots$
迭代次数 k X_2 和其近似向量 V_k $\tilde{X}_2$ 的相邻迭代误差 Wc λ_2 和其近似值 $lambda$	$k=2$ $X_2=(-0.250,-0.500,-1.000)^T$ $V_k=(-0.250,-0.500,-1.000)^T$ $Wc=6.252\ 343\ 455\ 491\ 521e-004$ $lambda=2.001\ 999\ 999\ 687\ 03$ $\lambda_2=1.999\ 999\ 999\ 999\ 99\cdots$	$k=2$ $X_2=(-0.250,-0.500,-1.000)^T$ $V_k=(-0.250,-0.500,-1.000)^T$ $Wc=3.131\ 258\ 787\ 820\ 493e-007$ $lambda=2.002\ 000\ 000\ 000\ 16$ $\lambda_2=1.999\ 999\ 999\ 999\ 99\cdots$
迭代次数 k X_3 和其近似向量 V_k $\tilde{X}_3$ 的相邻迭代误差 Wc λ_3 和其近似值 $lambda$	$k=2$ $X_3=(-0.400,-0.600,-1.000)^T$ $V_k=(0.400,0.600,1.000)^T$ $Wc=3.997\ 600\ 240\ 971\ 801e-004$ $lambda=4.001\ 999\ 999\ 800\ 30$ $\lambda_3=4.000\ 000\ 000\ 000\ 02\cdots$	$k=2$ $X_3=(-0.400,-0.600,-1.000)^T$ $V_k=(0.400,0.600,1.000)^T$ $Wc=1.996\ 005\ 395\ 108\ 913e-007$ $lambda=4.001\ 999\ 999\ 999\ 90$ $\lambda_3=4.000\ 000\ 000\ 000\ 02\cdots$

通过比较表 5-3 中列出的例题 5.3.3 和例题 5.3.4 的计算结果可见,由原点位移反幂法的迭代公式(5.27)的迭代序列的收敛速度比迭代公式(5.28)稍慢些.

(2) 再用两种原点位移反幂法的 MATLAB 主程序,求 $\tilde{\lambda}_1=0.999\ 999\ 999\ 999\ 97$ 对应的特征向量. 输入 MATLAB 程序

```
>>A=[0  11 -5; -2  17 -7; -4  26 -10]; V0=[1,1,1]';
[k,lambda,Vk,Wc]=ydwyfmf(A,V0,0.99999999999997,0.001,100)
```

运行后屏幕显示结果

请注意:因为 A-aE 的各阶主子式都不等于零,所以 A-aE 能进行 LU 分解.

A - aE 的秩 R(A - aE)和各阶顺序主子式值 hl、迭代次数 k,按模最小特征值的近似值 lambda,特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:

```
hl =
  -0.99999999999997   6.00000000000045   0.00000000000010
RA1  =                              Vk  =
    1.039168751049192e-013      0.50000000000000
k  =                            0.50000000000000
  2                             1.00000000000000
lambda  -                        Wc  =
       1.00000000000000          4.317692037236759e-013
```

输入 MATLAB 程序

```
>>A=[0  11  5;-2  17 -7;-4  26  10];V0=[1,1,1]';
[k,lambda,Vk,Wc]-wfmifa1(A,V0,0.99999999999997,0.001,100)
```

运行后屏幕显示结果

请注意迭代次数 k,特征值的近似值 lambda,对应的特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:

```
k  =                     Vk  =
  3                          0.50000000000000
lambda  =                    0.50000000000000
       1.00000000000000      1.00000000000000
Wc =
     5.412337245047640e-016
```

例 5.3.5 用原点位移反幂法的迭代公式(5.27),计算下列矩阵在给定的特征值的初始值的对应特征向量,精度为 0.001.

$$(1)\ A=\begin{pmatrix}1 & 2 & 2\\ 1 & -1 & 1\\ 4 & -12 & 1\end{pmatrix},\tilde{\lambda}_3=1.01;(2)\ A=\begin{pmatrix}1 & -1\\ 2 & 4\end{pmatrix},\tilde{\lambda}_2=2.001.$$

解 (1) 输入 MATLAB 程序

```
>>A=[1  2  2;1 -1  1;4 -12  1];V0=[1,1,1]';
[k,lambda,Vk,Wc]=wfmifa1(A,V0,1.01,0.001,100)
```

运行后屏幕显示结果

请注意迭代次数 k,特征值的近似值 lambda,特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:

```
k  -         lambda  =          Wc  =
  3                  1.0200       3.4622e-007,
V =
```

```
   -1.0000   -0.5280 -0.4720i   -0.5280 +0.4720i
   -0.3333   -0.1112 -0.1944i   -0.1112 +0.1944i
    0.3333    0.6112 +0.1664i    0.6112 -0.1664i
D =
  1.0000            0                       0
       0    -0.0000 +1.0000i                0
       0            0             -0.0000 -1.0000i
Vk =
   -1.0000   -0.3333  0.3333
```

(2) 输入 MATLAB 程序

```
>>A =[1 -1;2  4];V0 =[20,1]';
[k,lambda,Vk,Wc] =wfmifa1(A,V0,2.001,0.001,100)
```

运行后屏幕显示结果

请注意迭代次数 k,特征值的近似值 lambda,对应的特征向量的近似向量 Vk,相邻两次迭代的误差 Wc 如下:

```
k =          lambda =         Wc =              Vk =
   3                2.0020       0.280e-009     1        -1
V =                              D =
    -1.0000    0.5000              2       0
     1.0000   -1.0000              0       3
```



习 题 5.3

1. 用反幂法求矩阵 A 的对应特征值 $\tilde{\lambda}_3 = 1.267\,9$(精确特征值为 $\lambda_3 = 3-\sqrt{3}$)的特征向量的近似向量,用 5 个浮点数进行运算. 其中 $A=\begin{pmatrix}2&1&0\\1&3&1\\0&1&4\end{pmatrix}$.

2. 用反幂法计算下列矩阵的主特征值和对应的特征向量的近似向量,精度为 0.000 01.

(1) $\begin{pmatrix}-5&2&17\\2&-8&3\\3&3&-3\end{pmatrix}$;(2) $\begin{pmatrix}-11&2&15\\2&58&3\\15&3&-3\end{pmatrix}$.

5.4 雅可比方法及其 MATLAB 程序

雅可比方法是用来计算实对称矩阵的全部特征值和对应的特征向量的一种迭代的方法,最早由雅可比给出. 自从计算机出现以后,古典的雅可比方法已有

了不少的改进和推广.本节主要介绍雅可比方法及其 MATLAB 程序.

5.4.1 雅可比方法

雅可比方法是建立在下面定理的基础上的一种迭代方法.

定理 5.22 若 n 阶方阵 $\boldsymbol{A}$ 是实对称矩阵,则存在正交矩阵 $\boldsymbol{P}$,使得

$$\boldsymbol{P}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{P}=\operatorname{diag}(\lambda_1,\lambda_2,\cdots,\lambda_n)=\begin{pmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n\end{pmatrix},$$

其中 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 是 $\boldsymbol{A}$ 的全部特征值,而 $\boldsymbol{P}$ 的第 j 个列向量 $\boldsymbol{p}_j(j=1,2,\cdots,n)$ 是对应 $\lambda_j(f=1,2,\cdots,n)$ 的特征向量.

雅可比方法的思想是:根据定理 5.22,设法构造一系列简单的正交矩阵序列 $\boldsymbol{P}_1,\boldsymbol{P}_2,\cdots,\boldsymbol{P}_n,\cdots$,使得

$$\begin{cases}\boldsymbol{B}_0=\boldsymbol{A},\\ \boldsymbol{B}_k=\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{B}_{k-1}\boldsymbol{P}_k\end{cases}\quad(k=1,2,\cdots),\tag{5.29}$$

逐步地将实对称矩阵 $\boldsymbol{A}$ 化为对角矩阵,即

$$\lim_{k\to\infty}\boldsymbol{B}_k=\lim_{k\to\infty}\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{B}_{k-1}\boldsymbol{P}_k=\operatorname{diag}(\lambda_1,\lambda_2,\cdots,\lambda_n).\tag{5.30}$$

在实际情况下,当非对角线上的元接近零时,构造过程停止.则可得到

$$\boldsymbol{B}_k\approx\boldsymbol{B}=\operatorname{diag}(\lambda_1,\lambda_2,\cdots,\lambda_n).$$

另外,在构造正交矩阵序列 $\boldsymbol{P}_1,\boldsymbol{P}_2,\cdots,\boldsymbol{P}_n,\cdots$ 的过程中,产生

$$\begin{aligned}\boldsymbol{B}_k&=\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{B}_{k-1}\boldsymbol{P}_k=\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{P}_{k-1}^{\mathrm{T}}\boldsymbol{B}_{k-2}\boldsymbol{P}_{k-1}\boldsymbol{P}_k\\&=\cdots=\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{P}_{k-1}^{\mathrm{T}}\cdots\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{A}\boldsymbol{P}_1\cdots\boldsymbol{P}_{k-1}\boldsymbol{P}_k.\end{aligned}$$

记 $\boldsymbol{P}=\boldsymbol{P}_1\cdots\boldsymbol{P}_{k-1}\boldsymbol{P}_k$,则

$$\boldsymbol{P}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{P}=\boldsymbol{B}_k\approx\operatorname{diag}(\lambda_1,\lambda_2,\cdots,\lambda_n),\tag{5.31}$$

即

$$\boldsymbol{A}\boldsymbol{P}=\boldsymbol{P}\boldsymbol{B}_k\approx\boldsymbol{P}\operatorname{diag}(\lambda_1,\lambda_2,\cdots,\lambda_n).\tag{5.32}$$

令 $\boldsymbol{P}$ 的列向量用 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n$ 表示,则

$$\boldsymbol{P}=(\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n).\tag{5.33}$$

(5.32)式可表示为

$$(\boldsymbol{A}\boldsymbol{X}_1,\boldsymbol{A}\boldsymbol{X}_2,\cdots,\boldsymbol{A}\boldsymbol{X}_n)\approx(\lambda_1\boldsymbol{X}_1,\lambda_2\boldsymbol{X}_2,\cdots,\lambda_n\boldsymbol{X}_n).\tag{5.34}$$

由(5.34)式得

$$\boldsymbol{A}\boldsymbol{X}_i\approx\lambda_i\boldsymbol{X}_i(i=1,2,\cdots,n),\tag{5.35}$$

从而正交矩阵 $\boldsymbol{P}$ 的每个列向量 $\boldsymbol{X}_i(i=1,2,\cdots,n)$ 是 $\boldsymbol{A}$ 对应于特征值 λ_i 的特征向量的近似向量,以后简称为 λ_i 的近似特征向量.

5.4.2 实用的雅可比方法及其计算步骤

根据雅可比方法的思想,我们可以用以下步骤实现求 n 阶实对称矩阵 $\boldsymbol{A}$ 的特征值 λ_i 和对应于 λ_i 的近似特征向量 $\boldsymbol{X}_i$(预先给出 λ_i 和 $\boldsymbol{X}_i$ 的近似精度 ε).

(一)选取旋转主元所在的行数 i 和列数 j

因为雅可比方法的思想是:将实对称矩阵 $\boldsymbol{A}$ 化为对角矩阵,也就是通过迭代公式(5.29),逐次将非对角线上的元接近零.所以,古典的雅可比方法中,每一步变换之前都要寻查 $\boldsymbol{B}_k$ 的主对角线上方绝对值最大的元,作为它的旋转主元,这样编程序比较麻烦.这里我们分两步进行:

(1) 先用下面的 MATLAB 程序使 $\boldsymbol{B}_k$ 的主对角线上的元都为零,即

```
Ak = abs(Bk - diag(diag(Bk)));
```

(2) 再用下面的 MATLAB 程序寻找 $\boldsymbol{B}_k$ 的非主对角线上的元绝对值的最大者 $|a_{ij}|$,输出 $|a_{ij}|$ 及其 $|a_{ij}|$ 所在的行数 i 和列数 j,即

```
[m1 i] = max(Ak);
[m2 j] = max(m1); i = i(j).
```

(二)求使 B_k 的元 $b_{ij}=b_{ji}=0$ $(i\neq j)$ 的正交矩阵 P_k 的元

由 5.4.1 目中的讨论可见,雅可比迭代(5.29)中实对称矩阵 $\boldsymbol{B}_k=\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{B}_{k-1}\boldsymbol{P}_k$ 的每一步要使 $\boldsymbol{B}_k(k=1,2,\cdots,n,\cdots)$ 的两个非对角元 $b_{ij}=b_{ji}=0$ $(i\neq j,i,j=1,2,\cdots,n)$.设(5.29)中的

$$\boldsymbol{B}_1=\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{B}_0\boldsymbol{P}_1=\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{A}\boldsymbol{P}, \tag{5.36}$$

使两个非对角元 $b_{ij}=b_{ji}=0$ $(i\neq j,i,j=1,2,\cdots,n)$,其中 $\boldsymbol{P}_1$ 的形式为

$$\boldsymbol{P}_1=\begin{pmatrix}1&\cdots&0&\cdots&0&\cdots&0\\ \vdots&&\vdots&&\vdots&&\vdots\\ 0&\cdots&p_{ii}&\cdots&p_{ij}&\cdots&0\\ \vdots&&\vdots&&\vdots&&\vdots\\ 0&\cdots&-p_{ij}&\cdots&p_{ii}&\cdots&0\\ \vdots&&\vdots&&\vdots&&\vdots\\ 0&\cdots&0&\cdots&0&\cdots&1\end{pmatrix}\begin{matrix}\\ \\ \leftarrow\text{第 } i \text{ 行}\\ \\ \leftarrow\text{第 } j \text{ 行}\\ \\ \\ \end{matrix} \tag{5.37}$$

$$\qquad\qquad\uparrow\text{第 } i \text{ 列}\qquad\uparrow\text{第 } j \text{ 列}$$

这里除了位于第 i 行第 j 列上的元 $p_{ij}\neq 0$ 和第 j 列第 i 行上的元 $-p_{ij}\neq 0$ 以外,其余的非对角元 $p_{km}=0$ $(m\neq k,m,k\neq i,j,k,m=1,2,\cdots,n)$,并且除了位于第 i 行第 i 列上的元 p_{ii} 和第 j 列第 j 行上的元 p_{jj} 以外,其余的对角元 $p_{kk}=1$ $(k\neq i,j,k=1,2,\cdots,n)$.当 $p_{ii}=\cos\theta,p_{ij}=\sin\theta$ 时,这个 n 阶矩阵 $\boldsymbol{P}_1$ 为 $\mathbf{R}^n$ 中 x_i,x_j 平面内的一个平面旋转矩阵,θ 为旋转角,称 $\boldsymbol{A}$ 的(i,j)位置元 a_{ij} 为**旋转主元**.

$$
P_1 = P(i,j,\theta) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta & \cdots & \sin\theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & -\sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}_{n\times n} \begin{matrix} \\ \\ \leftarrow 第\ i\ 行 \\ \\ \leftarrow 第\ j\ 行 \\ \\ \\ \end{matrix} \tag{5.38}
$$

↑ 第 i 列 ↑ 第 j 列

下面验证(5.36)式只改变 $\boldsymbol{A}$ 第 i 行和第 j 行及第 i 列和第 j 列上的元. 设

$$
C_1 = AP_1 = \begin{pmatrix} a_{11} & \cdots & a_{1i} & \cdots & a_{1j} & \cdots & a_{nn} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{i1} & \cdots & a_{ii} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{j1} & \cdots & a_{ji} & \cdots & a_{jj} & \cdots & a_{jn} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{n1} & \cdots & a_{ni} & \cdots & a_{nj} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & p_{ii} & \cdots & p_{ij} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & -p_{ij} & \cdots & p_{ii} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}. \tag{5.39}
$$

计算(5.39)式,可以看出,(5.36)式只改变 $\boldsymbol{A}$ 的第 i 列和第 j 列上的元. 并且 $C_1 = (c_{ij})_{n\times n}$ 满足

$$
\begin{cases} c_{mk} = a_{mk}, & 当\ k\neq i, 且\ k\neq j, \\ c_{mi} = p_{ii}a_{mi} - p_{ij}a_{mj}, & m = 1,2,\cdots,n \\ c_{mj} = p_{ij}a_{mi} + p_{ii}a_{mj}, & m = 1,2,\cdots,n. \end{cases} \tag{5.40}
$$

例如,当取 n 阶矩阵 $\boldsymbol{P}_1$ 为 $\mathbf{R}^n$ 中 x_i, x_j 平面内的一个平面旋转矩阵(5.38)时,

$$
\begin{cases} c_{mk} = a_{mk}, & 当\ k\neq i, 且\ k\neq j, \\ c_{mi} = a_{mi}\cos\theta - a_{mj}\sin\theta, & m = 1,2,\cdots,n, \\ c_{mj} = a_{mi}\sin\theta + a_{mj}\cos\theta, & m = 1,2,\cdots,n. \end{cases}
$$

同理,$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{A}$ 只改变 $\boldsymbol{A}$ 的第 i 行和第 j 行上的元. 因此,(5.36)式只改变 $\boldsymbol{A}$ 第 i 行和第 j 行,第 i 列和第 j 列上的元. $\boldsymbol{B}_1 = (b_{ij})$ $(i,j = 1,2,\cdots,n)$ 的元 b_{ij} $(i,j = 1,2,\cdots,n)$ 可以用下列公式计算:

$$\begin{cases} b_{mi}=p_{ii}a_{mi}-p_{ij}a_{mj}, & \text{当 } m\neq i,j, \\ b_{mj}=p_{ij}a_{mi}+p_{ii}a_{mj}, & \text{当 } m\neq i,j, \\ b_{ii}=p_{ii}^2a_{ii}+p_{ij}^2a_{jj}-2p_{ii}p_{ij}a_{ij}, \\ b_{jj}=p_{ij}^2a_{ii}+p_{ii}^2a_{jj}+2p_{ii}p_{ij}a_{ij}, \\ b_{ij}=(p_{ii}^2-p_{ij}^2)a_{ij}+p_{ii}p_{ij}(a_{ii}-a_{jj}), \end{cases} \tag{5.41}$$

$\boldsymbol{B}_1$的其他元可根据对称性得到.

例如,当取 n 阶矩阵 $\boldsymbol{P}_1$ 为 $\mathbf{R}^n$ 中 x_i,x_j 平面内的一个平面旋转矩阵(5.38)时,对称矩阵 $\boldsymbol{B}_1=(b_{ij})$ $(i,j=1,2,\cdots,n)$ 的元 b_{ij} $(i,j=1,2,\cdots,n)$ 可以用下列公式计算:

$$\begin{cases} b_{mi}=a_{mi}\cos\theta-a_{mj}\sin\theta, & \text{当 } m\neq i,j, \\ b_{mj}=a_{mi}\sin\theta+a_{mj}\cos\theta, & \text{当 } m\neq i,j, \\ b_{ii}=a_{ii}\cos^2\theta+a_{jj}\sin^2\theta-2a_{ij}\cos\theta\sin\theta, \\ b_{jj}=a_{ii}\sin^2\theta+a_{jj}\cos^2\theta+2a_{ij}\cos\theta\sin\theta, \\ b_{ij}=(\cos^2\theta-\sin^2\theta)a_{ij}+(a_{ii}-a_{jj})\cos\theta\sin\theta, \end{cases}$$

即

$$\begin{cases} b_{mi}=a_{mi}\cos\theta-a_{mj}\sin\theta, & \text{当 } m\neq i,j, \\ b_{mj}=a_{mi}\sin\theta+a_{mj}\cos\theta, & \text{当 } m\neq i,j, \\ b_{ii}=a_{ii}\cos^2\theta+a_{jj}\sin^2\theta-a_{ij}\sin 2\theta, \\ b_{jj}=a_{ii}\sin^2\theta+a_{jj}\cos^2\theta+a_{ij}\sin 2\theta, \\ b_{ij}=a_{ij}\cos 2\theta+\dfrac{1}{2}(a_{ii}-a_{jj})\sin 2\theta. \end{cases} \tag{5.42}$$

雅可比迭代(5.29)中 $\boldsymbol{B}_k=\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{B}_{k-1}\boldsymbol{P}_k$ 的每一步要使 $\boldsymbol{B}_k(k=1,2,\cdots,n,\cdots)$ 的两个非对角元 $b_{ij}=b_{ji}=0$ $(i\neq j,i,j=1,2,\cdots,n)$,只要取(5.38)中的

$$\begin{cases} p_{ii}=\cos\theta, \\ p_{ij}=\sin\theta \end{cases} \tag{5.43}$$

即可. 由(5.42)中的第5式,得

$$\cot 2\theta=\frac{a_{jj}-a_{ii}}{2a_{ij}}\quad\left(|\theta|\leqslant\frac{\pi}{4}\right). \tag{5.44}$$

由(5.44)知,(1) 当 $a_{ii}=a_{jj}$时,$\cot 2\theta=0$,从而 $\theta=\pm\dfrac{\pi}{4}$;

(2) 当 $a_{ii}\neq a_{jj}$时,令 $\cot 2\theta=c$,$\tan\theta=t$,则从(5.44)和 $\tan 2\theta=\dfrac{2\tan\theta}{1-\tan^2\theta}=\dfrac{1}{c}$. 联立方程组,解得

$$t=\tan\theta=\frac{1}{c\pm\sqrt{1+c^2}}=\frac{\mathrm{sign}(c)}{|c|+\sqrt{1+c^2}},\tag{5.45}$$

$$\begin{cases}p_{ii}=\cos\theta=\dfrac{1}{\sqrt{1+\tan^2\theta}}=\dfrac{1}{\sqrt{1+t^2}},\\ p_{ij}=\sin\theta=\cos\theta\tan\theta=\dfrac{t}{\sqrt{1+t^2}}.\end{cases}\tag{5.46}$$

(三)控制迭代终止的条件

在每一轮消元时，都要给出一个控制量 ε_k，称为**控制迭代终止的条件**，也称为**消元容限**. 如果第 k 次迭代的矩阵 $\boldsymbol{B}_k$ 的非主对角线上的所有元的绝对值 $|a_{km}|<\varepsilon_k$，就迭代终止. 没有统一的迭代终止的条件，有的取 $\sum\limits_{\substack{i,j=1\\i\neq j}}^{n}|a_{ij}|<\delta$ 或者 $|a_{ij}|<\delta\sqrt{\dfrac{1}{n}\sum\limits_{i=1}^{n}|a_{ii}|^2}$ 时，迭代停止(其中 δ 是要求的精度)，效果不好. 在这里，选用预先给定 λ_i 和 $\boldsymbol{X}_i$ 的近似精度 $\varepsilon=\varepsilon_k$，当 $\boldsymbol{B}_k$ 的非主对角线上的元最大者 a_{ij} 的绝对值 $|a_{ij}|<\varepsilon$ 时，就迭代终止，效果非常好(请看例 5.4.1). 用下面的 MATLAB 程序作为控制迭代终止的条件，即

```
Wc = m2;
if(Wc > jd)
  state = 1;
else
  return
end
```

(四)计算正交矩阵 $\boldsymbol{P}=\boldsymbol{P}_1\boldsymbol{P}_2\cdots\boldsymbol{P}_{k-1}\boldsymbol{P}_k$

从(5.32)到(5.35)式可见，$\boldsymbol{P}$ 的列向量是所要求的特征向量 $\boldsymbol{X}_i(i=1,2,\cdots,n)$ 的近似向量. 所以，在第 k 次迭代停止时，需要计算出正交矩阵 $\boldsymbol{P}$. 这需要保存每一步迭代的 $\boldsymbol{V}_t=\boldsymbol{P}_1\boldsymbol{P}_2\cdots\boldsymbol{P}_{t-1}\boldsymbol{P}_t(t=1,2,\cdots,n)$. 在这里，我们取初始矩阵 $\boldsymbol{V}_0=\boldsymbol{E}$(其中 $\boldsymbol{E}$ 是单位矩阵)，再用 $\boldsymbol{V}_t=\boldsymbol{V}_{t-1}\boldsymbol{P}_t$ 保存. 设第 k 次迭代的矩阵 $\boldsymbol{V}_t^{(k)}=(\boldsymbol{v}_{ij}^{(k)})_{n\times n}$ 的元可以用下面的公式计算：

$$\begin{cases}v_{mi}^{(k)}=p_{ii}^{(k)}v_{mi}^{(k-1)}+p_{ij}^{(k)}v_{mj}^{(k-1)},\\ v_{mi}^{(k)}=-p_{ij}^{(k)}v_{mi}^{(k-1)}+p_{ii}^{(k)}v_{mj}^{(k-1)}, \quad (m=1,2,\cdots).\\ v_{mi}^{(k)}=v_{im}^{(k)} \quad (m\neq i,j)\end{cases}$$

综上所述，用雅可比迭代求 n 阶实对称矩阵 $\boldsymbol{A}$ 的特征值 λ_i 和对应于特征值 λ_i 的特征向量 $\boldsymbol{X}_i$ 的近似特征向量，精度为 ε 的一般步骤如下：

步骤 1 选旋转主元.

```
Ak = abs(Bk - diag(diag(Bk)));
[m1 i] = max(Ak);[m2 j] = max(m1);i = i(j);
```

步骤 2 求 $p_{ii}=\cos\theta$ 和 $p_{ij}=\sin\theta$

(1) 选择第 i 行和第 j 列,使得 $a_{ij}\neq 0$,求(5.46)式中的 $p_{ii}=\cos\theta$ 和 $p_{ij}=\sin\theta$.

当 $a_{jj}=a_{ii}$时,取 $\theta=\frac{\pi}{4}\text{sign}(a_{ij})$;

当 $a_{jj}\neq a_{ii}$时,计算

$$\begin{cases} c=\cot 2\theta=\dfrac{a_{jj}-a_{ii}}{2a_{ij}}, \\ t=\dfrac{\text{sign}(c)}{|c|+\sqrt{1+c^2}}, \\ p_{ii}=\dfrac{1}{\sqrt{1+t^2}}, \\ p_{ij}=\dfrac{t}{\sqrt{1+t^2}}. \end{cases} \tag{5.47}$$

(2) 构造 $\boldsymbol{B}=\boldsymbol{B}_1=(b_{ij})_{n\times n}$,其中

$$\begin{cases} b_{ij}=b_{ji}=0, \\ b_{mi}=b_{im}=p_{ii}a_{mi}-p_{ij}a_{mj}, & 当\ m\neq i,j, \\ b_{mj}=b_{jm}=p_{ij}a_{mi}+p_{ii}a_{mj}, & 当\ m\neq i,j, \\ b_{ii}=p_{ii}^2a_{ii}+p_{ij}^2a_{jj}-2p_{ii}p_{ij}a_{ij}, \\ b_{jj}=p_{ij}^2a_{ii}+p_{ii}^2a_{jj}+2p_{ii}p_{ij}a_{ij}. \end{cases} \tag{5.48}$$

步骤 3 计算第 k 次迭代的正交矩阵 $\boldsymbol{P}=\boldsymbol{P}_1\boldsymbol{P}_2\cdots\boldsymbol{P}_{k-1}\boldsymbol{P}_k$.

步骤 4 当 $\max\limits_{\substack{m\neq k \\ m,k=1,2,\cdots,n}}(|a_{mk}|)<\varepsilon$ 时,迭代停止,其中 ε 是要求的精度;否则,转到步骤 1、步骤 2 和步骤 3. 如此下去,当迭代 n 次后停止时,得到对称矩阵序列 $\boldsymbol{B}_n=\boldsymbol{P}_n^{\mathrm{T}}\boldsymbol{P}_{n-1}^{\mathrm{T}}\cdots\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{A}\boldsymbol{P}_1\cdots\boldsymbol{P}_{n-1}\boldsymbol{P}_n$ 和正交矩阵序列 $\boldsymbol{P}=\boldsymbol{P}_1\boldsymbol{P}_2\cdots\boldsymbol{P}_{n-1}\boldsymbol{P}_n$,则 $\boldsymbol{B}_n$ 的主对角线上元就是对称矩阵 $\boldsymbol{A}$ 的特征值的近似值 λ_i,$\boldsymbol{P}$ 的列向量就是特征值对应的特征向量 $\boldsymbol{X}_i(i=1,2,\cdots,n)$ 的近似向量,它们的精度为 ε.

例 5.4.1 用雅可比方法计算矩阵 $\boldsymbol{A}$ 的特征值 λ_i 和对应的特征向量 $\boldsymbol{X}_i$ $(i=1,2,3,4)$,其中 $\boldsymbol{A}=\begin{pmatrix} 12 & -56 & 3 & -1 \\ -56 & 7 & 2 & 0 \\ 3 & 2 & 5 & 1 \\ -1 & 0 & 1 & 12 \end{pmatrix}$.

解 取精度为 $\varepsilon=0.001$.

(1) 迭代次数 $k=1$ 时,计算如下:

① 选取旋转主元所在的行数 i 和列数 j

$\boldsymbol{A}$ 的非主对角线上的元的绝对值的最大者 $|a_{21}|=56$ 所在的行数 $i=2$ 和列数 $j=1$.

② 根据(5.47)式,求得 $p_{11}=p_{22}=\cos\theta=0.7227$,

$$p_{12}=-p_{21}=\sin\theta=0.6912,$$

且正交矩阵(即旋转矩阵)$\boldsymbol{P}_1$ 为

$$\boldsymbol{V}_1=\boldsymbol{P}_1=\begin{pmatrix}0.7227 & 0.6912 & 0 & 0\\ -0.6912 & 0.7227 & 0 & 0\\ 0 & 0 & 1.0000 & 0\\ 0 & 0 & 0 & 1.0000\end{pmatrix},$$

$$\boldsymbol{B}_1=\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{A}\boldsymbol{P}=\begin{pmatrix}65.5558 & 0 & 0.7858 & -0.7227\\ -0.0000 & -46.5558 & 3.5189 & -0.6912\\ 0.7858 & 3.5189 & 5.0000 & 1.0000\\ -0.7227 & -0.6912 & 1.0000 & 12.0000\end{pmatrix}.$$

③ 判断是否满足控制迭代终止的条件 $\max\limits_{\substack{m\neq k\\ m,k=1,2,\cdots,n}}(|a_{mk}|)<\varepsilon$.

因为第 $k=1$ 次迭代的矩阵 $\boldsymbol{B}_1$ 的非主对角线上的所有元的绝对值 $|a_{km}|$ 的最大值 $\max\limits_{\substack{m\neq k\\ m,k=1,2,\cdots,n}}(|a_{mk}|)=3.5189>\varepsilon$,所以继续迭代.

(2) 迭代次数 $k=2$ 时,计算如下:

① 选取旋转主元所在的行数 i 和列数 j.

$\boldsymbol{B}_1$ 的非主对角线上的元的绝对值的最大者 $|b_{32}|=3.5189$ 所在的行数 $i=3$ 和列数 $j=2$ 正交矩阵(即旋转矩阵)$\boldsymbol{P}_2$ 为

$$\boldsymbol{P}_2=\begin{pmatrix}1.0000 & 0 & 0 & 0\\ 0 & 0.9977 & 0.0678 & 0\\ 0 & -0.0678 & 0.9977 & 0\\ 0 & 0 & 0 & 1.0000\end{pmatrix}.$$

正交矩阵(即旋转矩阵)$\boldsymbol{V}_2=\boldsymbol{P}_1\boldsymbol{P}_2$ 为

$$\boldsymbol{V}_2=\begin{pmatrix}0.7227 & 0.6896 & 0.0468 & 0\\ -0.6912 & 0.7210 & 0.0490 & 0\\ 0 & -0.0678 & 0.9977 & 0\\ 0 & 0 & 0 & 1.0000\end{pmatrix},$$

$$\boldsymbol{B}_2=\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{B}_1\boldsymbol{P}_2=\begin{pmatrix}65.5558 & -0.0533 & 0.7840 & -0.7227\\ -0.0533 & -46.7949 & 0 & -0.7574\\ 0.7840 & 0.0000 & 5.2391 & 0.9509\\ -0.7227 & -0.7574 & 0.9509 & 12.0000\end{pmatrix}.$$

③ 因为第 $k=2$ 次迭代的矩阵 $\boldsymbol{B}_2$ 的非主对角线上的所有元的绝对值 $|a_{km}|$ 的最大值 $\max\limits_{\substack{m \neq k \\ m,k=1,2,\cdots,n}} (|a_{mk}|)=0.950\,9>\varepsilon$，所以继续迭代.

(3) 如此下去，经过 $k=10$ 次迭代，可得到，当矩阵 $\boldsymbol{B}_{10}$ 的非主对角线上的所有元的绝对值 $|a_{km}|$ 的最大值为 $Wc=6.920\,6\times10^{-4}$ 时，正交矩阵（即以特征向量为列向量的矩阵）$\boldsymbol{P}=\boldsymbol{V}_{10}=\boldsymbol{P}_1\boldsymbol{P}_2\cdots\boldsymbol{P}_9\boldsymbol{P}_{10}$，

$$\boldsymbol{V}_{10}=\begin{pmatrix} 0.722\,9 & 0.689\,9 & 0.037\,3 & 0.005\,9 \\ -0.690\,7 & 0.720\,6 & 0.060\,0 & -0.010\,3 \\ 0.012\,8 & -0.068\,0 & 0.988\,1 & 0.137\,36 \\ -0.013\,3 & 0.012\,9 & -0.136\,6 & 0.990\,5 \end{pmatrix},$$

即

$$\boldsymbol{V}_{10}=\begin{pmatrix} 0.723 & 0.690 & 0.037 & 0.006 \\ -0.691 & 0.721 & 0.060 & -0.010 \\ 0.013 & -0.068 & 0.988 & 0.137\,4 \\ -0.013 & 0.013 & -0.137 & 0.991 \end{pmatrix},$$

$$\boldsymbol{B}_{10}=\boldsymbol{P}_{10}^{\mathrm{T}}\boldsymbol{B}_9\boldsymbol{P}_{10}=\begin{pmatrix} 65.575\,4 & 0.000\,0 & -0.000\,2 & 0.000\,0 \\ -0.000\,0 & -46.804\,6 & -0.000\,0 & -0.000\,7 \\ -0.000\,2 & -0.000\,0 & 5.095\,4 & 0.007\,5 \\ 0.000\,0 & -0.000\,7 & 0.007\,5 & 12.133\,8 \end{pmatrix},$$

$\boldsymbol{B}_{10}$ 的主对角线上的元，即特征值的近似值的向量为

$$\boldsymbol{D}_{10}=(65.575\,4 \quad -46.804\,6 \quad 5.095\,4 \quad 12.133\,8)^{\mathrm{T}}.$$

其实，$\boldsymbol{A}$ 的精确的特征值的向量为

$$\boldsymbol{\lambda}=(65.575\,4 \quad -46.804\,6 \quad 5.095\,4 \quad 12.133\,8)^{\mathrm{T}}.$$

$\boldsymbol{B}_{10}$ 与 $\boldsymbol{\lambda}$ 的各个分量的最大绝对误差为 $6.936\,3\times10^{-10}$.

$$\boldsymbol{X}=\begin{pmatrix} -0.722\,9 & 0.689\,9 & -0.037\,3 & 0.005\,9 \\ 0.690\,7 & 0.720\,6 & -0.060\,0 & -0.010\,3 \\ -0.012\,8 & -0.068\,0 & -0.988\,0 & 0.138\,4 \\ 0.013\,3 & 0.012\,9 & 0.137\,7 & 0.990\,3 \end{pmatrix},$$

即

$$\boldsymbol{X}=\begin{pmatrix} -0.723 & 0.690 & -0.037 & 0.006 \\ 0.691 & 0.721 & -0.060 & -0.010 \\ -0.013 & -0.068 & -0.988 & 0.138 \\ 0.013 & 0.013 & 0.138 & 0.990 \end{pmatrix}.$$

$\boldsymbol{V}_{10}$ 与 $\boldsymbol{X}$ 的各个分量的最大绝对误差为 0.001.

5.4.3 雅可比方法的 MATLAB 程序

设 n 阶实对称矩阵 A 的 n 个特征值为 $\lambda_1,\lambda_2,\cdots,\lambda_n$,特征值 λ_i 对应的特征向量为 X_i,则我们可以用下面的 MATLAB 程序计算 A 的全部特征值和对应的特征向量的近似值和向量.

用雅可比方法计算对称矩阵 A 的特征值和对应的特征向量的 MATLAB 主程序

输入的量:n 阶实对称矩阵 A、计算的精度 jd、迭代的最大次数 $max1$;

输出的量:计算全过程中每次迭代的序号 k,选取的旋转主元 mk 及其所在的行数 i 和列数 j,c,t,p_{ii} 和 p_{ij} 的值,旋转矩阵 P_k,正交矩阵 $V_k=P_1P_2\cdots P_k$,对称矩阵 B_k,判断是否满足控制迭代终止的条件 Wc,以特征向量为列向量的矩阵 V,特征值为对角元的对角矩阵 D. 如果迭代次数已经达到最大的迭代次数 $max1$,则给出提示的相关信息.

根据雅可比迭代求 n 阶实对称矩阵 A 的特征值 λ_i 和对应的特征向量 X_i 的近似向量,精度为 ε 的一般步骤,现提供 MATLAB 主程序如下:

```
function[k,Bk,V,D,Wc] = jacobite(A,jd,max1)
[n,n] = size(A);Vk = eye(n);Bk = A;state = 1;k = 0;P0 = eye(n);
Aij = abs(Bk - diag(diag(Bk)));[m1 i] = max(Aij);
[m2 j] = max(m1);i = i(j);
while((k < = max1)&(state = = 1))
k = k + 1,aij = abs(Bk - diag(diag(Bk)));[m1 i] = max(abs(aij));
[m2 j] = max(m1);i = i(j),j,Aij = (Bk - diag(diag(Bk)));mk = m2 *
sign(Aij(i,j)),
Wc = m2,Dk = diag(diag(Bk));Pk = P0;c = (Bk(j,j) - Bk(i,i))/(2 * Bk
(i,j)),
t = sign(c)/(abs(c) + sqrt(1 + c^2)),
pii = 1/(sqrt(1 + t^2)),pij = t/(sqrt(1 + t^2)),Pk(i,i) = pii;Pk(i,
j) = pij;
Pk(j,j) = pii;Pk(j,i) = - pij;Pk,B1 = Pk' * Bk;B2 = B1 * Pk;Vk = Vk *
Pk,Bk = B2,
if(Wc > jd)
  state = 1;
  else
      return
  end
  Pk;Vk;Bk = B2;Wc;
```

```
end
if(k>max1)
  disp('请注意迭代次数 k 已经达到最大迭代次数 max1,迭代次数 k,对称矩阵 Bk,以特征向量为列向量的矩阵 V,特征值为对角元的对角矩阵 D 如下:')
  else
  disp('请注意迭代次数 k,对称矩阵 Bk,以特征向量为列向量的矩阵 V,特征值为对角元的对角矩阵 D 如下:')
end
Wc;k=k;V=Vk;Bk=B2;D=diag(diag(Bk));
[V1,D1]=eig(A,'nobalance')
```

例 5.4.2 用雅可比方法的 MATLAB 程序计算例 5.4.1 中矩阵 $\boldsymbol{A}$ 的特征值 λ_i 和对应的特征向量 $\boldsymbol{X}_i(i=1,2,3,4)$.

解 (1) 保存名为 jacobite.m 为 M 文件;

(2) 输入 MATLAB 程序

```
>>A-[12 -56  3 -1;-56  7  2  0;3  2  5  1;-1  0  1  1  2];
[k,B,V,D,Wc]=jacobite(A,0.001,100)
```

(3) 运行后屏幕显示计算的全过程中每次迭代的序号 k,选取的旋转主元 mk 及其所在的行数 i 和列数 j,c,t,p_{ii} 和 p_{ij} 的值,旋转矩阵 $\boldsymbol{P}_k$,正交矩阵 $\boldsymbol{V}_k=\boldsymbol{P}_1\boldsymbol{P}_2\cdots\boldsymbol{P}_k$,对称矩阵 $\boldsymbol{B}_k$,判断是否满足控制迭代终止的条件 Wc,以特征向量为列向量的矩阵 $\boldsymbol{V}$,特征值为对角元的对角矩阵 $\boldsymbol{D}$ 和相关信息如下:

```
k =          i =          j =          mk =        Wc =
 1            2            1            -56          56
c =                                   t =
  -0.04464285714286                     -0.95635313919972
pii =                                 pij =
     0.72270271801843                   -0.69115901308510
Pk =
    0.72270271801843   0.69115901308510                  0                  0
   -0.69115901308510   0.72270271801843                  0                  0
                   0                  0   1.00000000000000                  0
                   0                  0                  0   1.00000000000000
Vk =
    0.72270271801843   0.69115901308510                  0                  0
   -0.69115901308510   0.72270271801843                  0                  0
                   0                  0   1.00000000000000                  0
                   0                  0                  0   1.00000000000000
Bk =
```

```
    65.55577579518456                    0   0.78579012788509  -0.72270271801843
   -0.00000000000001  -46.55577579518456   3.51888247529217  -0.69115901308510
    0.78579012788509    3.51888247529217   5.00000000000000   1.00000000000000
   -0.72270271801843   -0.69115901308510   1.00000000000000  12.00000000000000
k =    i =    j =    mk =                 Wc =
  2      3      2    3.51888247529217      3.51888247529217
c =                          t =
   -7.32558932518824          -0.06793885568129
pii =                        pij =
  0.99770011455446            -0.06778260409592
Pk =
  1.00000000000000         0                  0                  0
         0    0.99770011455446   0.06778260409592                  0
         0   -0.06778260409592   0.99770011455446                  0
         0                  0                  0   1.00000000000000
Vk =
 0.72270271801843   0.68956942653035 0.04684855775127                  0
-0.69115901308510   0.72104058455581 0.04898667221449                  0
                0  -0.06778260409592 0.99770011455446                  0
                0                  0                0  1.00000000000000
Bk =
65.55577579518456   -0.05326290114092  0.78398290060672  -0.72270271801843
-0.05326290114093  -46.79484464383285                 0  -0.75735203062627
 0.78398290060672    0.00000000000000  5.23906884864829   0.95085155680318
-0.72270271801843   -0.75735203062627  0.95085155680318  12.00000000000000
k =    i =     j =     mk =                Wc =
  3      4       3     0.95085155680318    0.95085155680318
c = t =
  -3.55519802380213   -0.13796227443116
pii =                        pij =
   0.99061693994324            -0.13666776612460
Pk =
1.00000000000000                 0                 0                 0
               0  1.00000000000000                 0                 0
               0                 0  0.99061693994324  0.13666776612460
               0                 0 -0.13666776612460  0.99061693994324
```

```
      Vk =
 0.72270271801843    0.68956942653035    0.04640897492032  0.00640268773403
-0.69115901308510    0.72104058455581    0.04852702732712  0.00669489906143
                0   -0.06778260409592    0.98833863446096  0.13635344591842
                0                   0   -0.13666776612460  0.99061693994324
      Bk =
65.55577579518456   -0.05326290114092    0.87539690801061  -0.60877636330628
-0.05326290114093  -46.79484464383285    0.10350561019562  -0.75024575103880
 0.87539690801061    0.10350561019562    5.10788720522532  -0.00000000000000
-0.60877636330628   -0.75024575103880   -0.00000000000000  12.13118164342297
      k =     i =     j =    mk =                  Wc =
       4        1      3       0.87539690801061   0.87539690801061
      c =                         t =
        -34.52598931799430          -0.01447880833914
      pii =                   pij =
          0.99989519853186        -0.01447729093877
      Pk =
0.99989519853186                      0  -0.01447729093877                 0
               0 1.00000000000000                         0                 0
0.01447729093877                      0   0.99989519853186                 0
               0                      0                   0 1.00000000000000
      Vk =
 0.72329885394465    0.68956942653035    0.03594133368062  0.00640268773403
-0.69038403871280    0.72104058455581    0.05852805174080  0.00669489906143
 0.01430846595712   -0.06778260409592    0.98823505512105  0.13635344591842
-0.00197857901214                   0   -0.13665344314206  0.99061693994324
      Bk =
65.56845049923633   -0.05175883827808   -0.00000000000000  -0.60871256264964
-0.05175883827809  -46.79484464383285    0.10426586517177  -0.75024575103880
-0.00000000000000    0.10426586517177    5.09521250117356   0.00881343252823
-0.60871256264964   -0.75024575103880    0.00881343252823  12.13118164342297
      k =     i =     j =    mk =                      Wc =
       5        4      2       -0.75024575103880   0.75024575103880
      c =                     t =
        39.27114962375084      0.01272992971264
      pii =                   pij =
          0.99991898429114        0.01272889838836
      Pk =
```

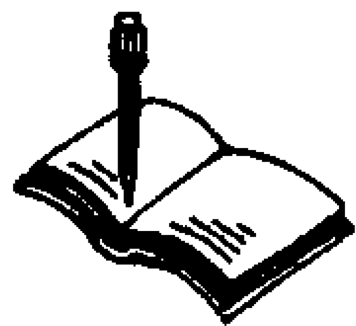
```
1.00000000000000                   0                   0                   0
               0   0.99991898429114                    0  -0.01272889838836
               0                   0    1.00000000000000                   0
               0   0.01272889838836                    0   0.99991898429114
      Vk =
 0.72329885394465   0.68959505973603   0.03594133368062  -0.00237529014628
-0.69038403871280   0.72106738763160   0.05852805174080  -0.00248369566525
 0.01430846595712  -0.06604148348220   0.98823505512105   0.13720519702737
-0.00197857901214   0.01260946237032   0.13665344314206   0.99053668440964
      Bk =
65.56845049923633  -0.05950288535679  -0.00000000000000  -0.60800441437674
-0.05950288535680 -46.80439521951078   0.10436960328590   0.00000000000000
-0.00000000000000   0.10436960328590   5.09521250117356   0.00748552889860
-0.60800441437674   0.00000000000000   0.00748552889860  12.14073221910090
      k =     i =      j =     mk =                  Wc =
       6        4        1      -0.60800441437674   0.60800441437674
      c =                      t =
       -43.93694931878409    -0.01137847012503
      pii =                    pij =
          0.99993527149402        -0.01137773361366
      Pk =
 0.99993527149402                  0                   0   0.01137773361366
                0   1.00000000000000                   0                  0
                0                  0    1.00000000000000                  0
-0.01137773361366                  0                   0   0.99993527149402
      Vk -
 0.72327906130899   0.68959505973603   0.03594133368062   0.00585436528595
-0.69031109235777   0.72106738763160   0.05852805174080  -0.01033854058294
 0.01274645560931  -0.06604148348220   0.98823505512105   0.13735911385404
-0.01324851347145   0.01260946237032  -0.13665344314206   0.99045005670500
      Bk =
65.57536865930122  -0.05949903382392  -0.00008516835377  -0.00000000000000
-0.05949903382393 -46.80439521951078   0.10436960328590  -0.00067700797883
-0.00008516835377   0.10436960328590   5.09521250117356   0.00748504437150
-0.00000000000000  -0.00067700797883   0.00748504437150  12.13381405903603
      k =     i =      j =       mk =                  Wc =
       7        3        2         0.10436960328590   0.10436960328590
      c =                      t =
```

```
   -2.486337309269764e+002    -0.00201098208240
pii =                        pij =
   0.99999797798167 -0.00201097801616
Pk =
1.00000000000000                  0                  0                  0
               0   0.99999797798167 0.00201097801616                  0
               0  -0.00201097801616 0.99999797798167                  0
               0                  0                  0 1.00000000000000
```

请注意迭代次数 k,对称矩阵 Bk,以特征向量为列向量的矩阵 V,特征值为对角元的对角矩阵 D 如下:

```
V1 =
  0.68990429476497  -0.03732423222484   0.00588594854431  -0.72291377173450
  0.72058252860300  -0.05998661236737  -0.01028322161977   0.69069289931337
 -0.06802029759277  -0.98795368410472   0.13841044442471  -0.01277912569225
  0.01288885768193   0.13768088498200   0.99030407443219   0.01325486405899
D1 =
-46.80463661419736                  0                  0                  0
                 0 5.09541442877727                  0                  0
                 0                  0 12.13382202426702                  0
                 0                  0                  0 65.57540016115307
k =
   10
B =
 65.57540016045945   0.00000000000175  -0.00020481967566   0.00000014862836
  0.00000000000175 -46.80463661419739   0.00000062739984   0.00000000000000
 -0.00020481967566   0.00000062739984   5.09541442947090  -0.00000000000737
  0.00000014862836  -0.00000000000000  -0.00000000000737  12.13382202426704
V =
  0.72291389811507   0.68990429521617   0.03732177568689   0.00588595055487
 -0.69069269613201   0.72058252932816   0.05998894273570  -0.01028322354062
  0.01278247108107  -0.06802028564977   0.98795364164379   0.13841044446122
 -0.01325533307898   0.01288885601755  -0.13768084024946   0.99030407439520
D =
65.57540016045945                  0                  0                  0
                0 -46.80463661419739                  0                  0
                0                  0 5.09541442947090                  0
                0                  0                  0 12.13382202426704
```

```
Wc =
   6.920584967017158e-004
```



习 题 5.4

1. 用雅可比方法计算矩阵 A 的特征值 λ_i 和对应的特征向量 $X_i(i=1,2,3,4)$,其中

$$A=\begin{pmatrix} 0 & 12 & 16 & -15 \\ 12 & 288 & 309 & 185 \\ 16 & 309 & 312 & 80 \\ -15 & 185 & 80 & -600 \end{pmatrix}.$$

2. 用两种雅可比方法计算矩阵 A 的特征值 λ_i 和对应的特征向量 X_i $(i=1,2,3,4)$,其中

$$A=\begin{pmatrix} 8 & -1 & 3 & -1 \\ -1 & 6 & 2 & 0 \\ 3 & 2 & 9 & 1 \\ -1 & 0 & 1 & 7 \end{pmatrix}.$$

5.5 豪斯霍尔德(Householder)方法及其 MATLAB 程序

在雅可比方法中,每一次迭代产生两个值为零的非对角元,但接下来的迭代使它们变成非零. 因此需要许多次迭代才能使得非对角元足够接近零. 豪斯霍尔德方法可以在一次迭代中产生多个值为零的非对角元,而且在接下来的迭代中使它们保持为零. 本节介绍豪斯霍尔德方法及其 MATLAB 程序.

5.5.1 豪斯霍尔德方法及其 MATLAB 程序

豪斯霍尔德方法是建立在下面理论的基础上的一种迭代方法.

定义 5.7 设 n 阶方阵 $A=(a_{ij})_{n\times n}$,如果当 $i>j+1$ 时有 $a_{ij}=0$,则称 A 为**上豪斯霍尔德矩阵**,即

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ & \ddots & \ddots & \vdots \\ & & a_{n,n-1} & a_{nn} \end{pmatrix}$$

定义 5.8 设向量 ω 满足 $\|\omega\|_2=1$,则矩阵

$$P=E-2\omega\omega^{\mathrm{T}}$$

称为**初等反射矩阵**,记为 $P(\omega)$,即

$$P(\boldsymbol{\omega})=\begin{pmatrix}1-2\omega_1^2 & -2\omega_1\omega_2 & \cdots & -2\omega_1\omega_n\\ -2\omega_2\omega_1 & 1-2\omega_2^2 & \cdots & -2\omega_2\omega_n\\ \vdots & \vdots & & \vdots\\ -2\omega_n\omega_1 & -2\omega_n\omega_2 & \cdots & 1-2\omega_n^2\end{pmatrix},$$

其中 $\boldsymbol{\omega}=(\omega_1,\omega_2,\cdots,\omega_n)^{\mathrm{T}}$.

定理 5.23 若 n 阶方阵 $\boldsymbol{A}$ 是初等反射矩阵,则 $\boldsymbol{A}$ 是对称矩阵($\boldsymbol{A}^{\mathrm{T}}=\boldsymbol{A}$)、正交矩阵($\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}=\boldsymbol{E}$)和对合矩阵($\boldsymbol{A}^2=\boldsymbol{E}$).

定理 5.24 若 $\boldsymbol{X},\boldsymbol{Y}$ 是两个不相等的 n 维向量,且 $\|\boldsymbol{X}\|_2=\|\boldsymbol{Y}\|_2$,则存在一个初等反射变换 $P(\boldsymbol{\omega})$,使得 $\boldsymbol{PX}=\boldsymbol{Y}$.

推论 5.2 若 n 维向量 $\boldsymbol{X}\neq\boldsymbol{O},\sigma=\pm\|\boldsymbol{X}\|_2$,且 $\boldsymbol{X}\neq-\sigma\boldsymbol{e}_1$,则存在一个初等反射矩阵

$$\boldsymbol{P}=\boldsymbol{E}-2\frac{\boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}}{\|\boldsymbol{\mu}\|_2^2}, \tag{5.49}$$

使得 $\boldsymbol{PX}=-\sigma\boldsymbol{e}_1$,其中 $\boldsymbol{\mu}=\boldsymbol{X}+\sigma\boldsymbol{e}_1$.

设 n 维向量 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}\neq\boldsymbol{O},\boldsymbol{\mu}=(u_1,u_2,\cdots,u_n)^{\mathrm{T}}$,则

$$\boldsymbol{\mu}=\boldsymbol{X}+\sigma\boldsymbol{e}_1=(\sigma+x_1,x_2,\cdots,x_n)^{\mathrm{T}}, \tag{5.50}$$

$$\frac{\|\boldsymbol{\mu}\|_2^2}{2}=\frac{1}{2}[(\sigma+x_1)^2+x_2^2+\cdots+x_n^2]=\sigma(x_1+\sigma). \tag{5.51}$$

如果 σ 和 x_1 异号,则计算 $x_1+\sigma$ 时,有效数字可能损失,所以我们取 σ 和 x_1 同号,即

$$\sigma=\|\boldsymbol{X}\|_2\operatorname{sign}(x_1). \tag{5.52}$$

设向量 $\boldsymbol{X}_{n\times1}\neq\boldsymbol{O}$,确定一个 n 阶初等反射矩阵 $\boldsymbol{P}_n$,使得 $\boldsymbol{P}_n\boldsymbol{X}_{n\times1}$ 的后 $n-1$ 个分量都为零,则我们可以用下面的 MATLAB 程序计算.

求初等反射矩阵 $\boldsymbol{P}$,使得 $\boldsymbol{PX}$ 的第一个分量以外的其余的分量都为零的 MATLAB 主程序

输入的量:n 维向量 $\boldsymbol{X}\neq\boldsymbol{O}$;

输出量:迭代次数 k,对称矩阵 $\boldsymbol{B}$,以特征向量为列向量的矩阵 $\boldsymbol{V}$,特征值为对角元的对角矩阵 $\boldsymbol{D}$. 如果迭代次数已经达到最大的迭代次数 $max1$,则给出提示的相关信息.

根据(5.49)至(5.52)式编写 MATLAB 主程序如下:

```
function [xigema,rou,miou,P,PX] = Householder(X)
n = size(X);nX = norm(X,2);
xigema = nX * sign(X(1));
rou = xigema * (xigema + X(1));
```

```
miou=[xigema,zeros(1,n-1)]'+X,
E=eye(n,n); C=2*miou*(miou)';
P=E-C/(norm(miou,2)^2); PX=P*X;
```

例 5.5.1 设向量 $\boldsymbol{X}=(2,2,1)^{\mathrm{T}}$,确定一个初等反射矩阵 $\boldsymbol{P}$,使得 $\boldsymbol{PX}$ 的后两个分量为零.

解 方法 1 因为 $\sigma=\|\boldsymbol{X}\|_2\operatorname{sign}(x_1)=\sqrt{2^2+2^2+1^2}\operatorname{sing}(2)=3$,

$$\rho=\sigma(x_1+\sigma)=3(2+3)=15,$$

$$\boldsymbol{\mu}=\boldsymbol{X}+\sigma\boldsymbol{e}_1=(\sigma+x_1,x_2,x_3)^{\mathrm{T}}=(3+2,2,1)^{\mathrm{T}}=(5,2,1)^{\mathrm{T}},$$

所以 $\boldsymbol{P}=\boldsymbol{E}-2\dfrac{\boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}}{\|\boldsymbol{\mu}\|_2^2}=\begin{pmatrix}-2/3 & -2/3 & -1/3\\ -2/3 & 11/15 & -2/15\\ -1/3 & -2/15 & 14/15\end{pmatrix}$,

且 $\boldsymbol{PX}=(-3,1/1441151880758559,1/2251799813685248)^{\mathrm{T}}$,即 $\boldsymbol{PX}=(-3,0,0)^{\mathrm{T}}$.

方法 2 (1) 保存名为 Householder.m 为 M 文件;

(2) 输入 MATLAB 程序

```
>> X=[2 2 1]'; [xigema,rou,miou,P,PX]=Householder(X)
```

(3) 运行后屏幕显示结果

```
P =                                             PX =
    -0.6667   -0.6667   -0.3333                      -3.0000
    -0.6667    0.7333   -0.1333                       0.0000
    -0.3333   -0.1333    0.9333                       0.0000
```

5.5.2 矩阵约化为上豪斯霍尔德矩阵及其 MATLAB 程序

我们讨论的问题是用正交相似变换(即豪斯霍尔德变换)约化一般的矩阵 $\boldsymbol{A}$ 为上豪斯霍尔德矩阵.

定理 5.25 设 $\boldsymbol{A}$ 是 n 阶实矩阵,则存在 $n-2$ 个初等反射矩阵 $\boldsymbol{U}_1,\boldsymbol{U}_2,\cdots,\boldsymbol{U}_{n-2}$,使得 $\boldsymbol{A}_{n-1}=\boldsymbol{U}_{n-2}\cdots\boldsymbol{U}_2\boldsymbol{U}_1\boldsymbol{A}$ 为上豪斯霍尔德矩阵,即

$$\boldsymbol{A}_{n-1}=\begin{pmatrix}a_{11} & a_{12}^{(2)} & a_{13}^{(3)} & \cdots & a_{1n}^{(n-1)}\\ -\sigma_1 & a_{22}^{(2)} & a_{23}^{(3)} & \cdots & a_{2n}^{(n-1)}\\ & -\sigma_2 & a_{33}^{(3)} & \cdots & a_{3n}^{(n-1)}\\ & & \ddots & \ddots & \vdots\\ & & & -\sigma_{n-1} & a_{nn}^{(n-1)}\end{pmatrix}.$$

证明 将 n 阶实矩阵 $\boldsymbol{A}$ 分块为

$$A=\left(\begin{array}{c|ccc} a_{11} & a_{12} & \cdots & a_{1n} \\ \hline a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{array}\right)=\begin{pmatrix} a_{11} & A_{12}^{(1)} \\ A_{21}^{(1)} & A_{22}^{(1)} \end{pmatrix}.$$

下面用数学归纳法证明.

(1) 不妨设 $A_{21}^{(1)}\neq O$,否则这一步不需要约化. 选择初始反射矩阵 P_1,使得 $P_1A_{21}^{(1)}=-\sigma_1e_1$,其中

$$\begin{cases} \sigma_1=\|A_{21}^{(1)}\|_2\operatorname{sign}(a_{21}), \\ \mu_1=A_{21}^{(1)}+\sigma_1e_1, \\ c_1=\dfrac{1}{2}\|\mu_1\|_2^2=\sigma_1(a_{21}+\sigma_1), \\ P_1=E-\dfrac{\mu_1\mu_1^{\mathrm T}}{c_1}. \end{cases} \tag{5.53}$$

令 $A_1=A$,且 $U_1=\begin{pmatrix} E_1 & O \\ O & P_1 \end{pmatrix}$,则

$$A_2=U_1A_1=\begin{pmatrix} a_{11} & A_{12}^{(1)} \\ P_1A_{21}^{(1)} & P_1A_{22}^{(1)} \end{pmatrix}=\begin{pmatrix} A_{11}^{(2)} & A_{12}^{(2)} & A_{13}^{(2)} \\ O & A_{22}^{(2)} & A_{23}^{(2)} \end{pmatrix}$$

$$=\begin{pmatrix} a_{11} & a_{12}^{(2)} & a_{13}^{(2)} & \cdots & a_{1n}^{(2)} \\ -\sigma_1 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix},$$

其中 $A_{11}^{(2)}\in\mathbf{R}^{2\times 1}$,$A_{22}^{(2)}\in\mathbf{R}^{n-2}$,$A_{23}^{(2)}\in\mathbf{R}^{(n-2)\times(n-2)}$.

(2) 仿照(1),不妨设 $n-2$ 维列向量 $A_{22}^{(2)}=(a_{32}^{(2)},a_{42}^{(2)},\cdots,a_{n2}^{(2)})^{\mathrm T}\neq O$,否则这一步不需要约化. 选择初始反射矩阵 P_2,使得 $P_2A_{22}^{(2)}=-\sigma_2e_1$,其中

$$\begin{cases} \sigma_2=\|A_{22}^{(2)}\|_2\operatorname{sign}(a_{32}^{(2)}), \\ \mu_2=A_{22}^{(2)}+\sigma_2e_1, \\ c_2=\dfrac{1}{2}\|\mu_2\|_2^2=\sigma_2(a_{32}^{(2)}+\sigma_2), \\ P_2=E-\dfrac{\mu_2\mu_2^{\mathrm T}}{c_2}. \end{cases}$$

令 $U_2=\begin{pmatrix} E_2 & O \\ O & P_2 \end{pmatrix}$,则

$$A_3=U_2A_2=\begin{pmatrix}A_{11}^{(3)} & A_{12}^{(3)} & A_{13}^{(3)}\\ O & A_{22}^{(3)} & A_{23}^{(3)}\end{pmatrix}=\begin{pmatrix}a_{11} & a_{12}^{(2)} & a_{13}^{(3)} & \cdots & a_{1n}^{(3)}\\ -\sigma_1 & a_{22}^{(2)} & a_{23}^{(3)} & \cdots & a_{2n}^{(3)}\\ 0 & -\sigma_2 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)}\\ 0 & 0 & a_{43}^{(3)} & \cdots & a_{4n}^{(3)}\\ \vdots & \vdots & \vdots & & \vdots\\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)}\end{pmatrix},$$

其中 $A_{11}^{(3)}\in\mathbf{R}^{3\times 2}, A_{22}^{(3)}\in\mathbf{R}^{n-3}, A_{23}^{(3)}\in\mathbf{R}^{(n-3)\times(n-3)}$.

如此继续下去,由数学归纳法可以证明,经过对 A 进行 $k-1$ 步正交相似约化,A_k 具有如下形式

$$A_k=U_{k-1}A_{k-1}=\begin{pmatrix}A_{11}^{(k)} & A_{12}^{(k)} & A_{13}^{(k)}\\ O & A_{22}^{(k)} & A_{23}^{(k)}\end{pmatrix},$$

其中 $A_{11}^{(k)}\in\mathbf{R}^{k\times(k-1)}, A_{22}^{(k)}\in\mathbf{R}^{n-k}, A_{23}^{(k)}\in\mathbf{R}^{(n-k)\times(n-k)}$.

设列向量 $A_{22}^{(k)}\neq O$,选择初始反射矩阵 P_k,使得 $P_kA_{22}^{(k)}=-\sigma_k e_1$,其中

$$\begin{cases}\sigma_k=\|A_{22}^{(k)}\|_2\operatorname{sign}(a_{k+1,k}^{(k)}),\\ \mu_k=A_{22}^{(k)}+\sigma_k e_1,\\ c_k=\dfrac{1}{2}\|\mu_k\|_2^2=\sigma_k(a_{k+1,k}^{(k)}+\sigma_k),\\ P_k=E-\dfrac{\mu_k\mu_k^{\mathrm{T}}}{c_k}.\end{cases}\tag{5.54}$$

令 $U_k=\begin{pmatrix}E_k & O\\ O & P_k\end{pmatrix}$,则

$$A_{k+1}=U_kA_k=\begin{pmatrix}A_{11}^{(k)} & A_{12}^{(k)} & A_{13}^{(k)}\\ O & P_kA_{22}^{(k)} & P_kA_{23}^{(k)}\end{pmatrix}=\begin{pmatrix}A_{11}^{(k)} & A_{12}^{(k)} & A_{13}^{(k)}\\ O & -\sigma_k e_1 & P_kA_{23}^{(k)}\end{pmatrix}.\tag{5.55}$$

当 $k=n-2$ 时,(5.55)式为上豪斯霍尔德矩阵:

$$A_{n-1}=U_{n-2}A_{n-2}=U_{n-2}\cdots U_2U_1A,$$

即

$$A_{n-1}=\begin{pmatrix}a_{11} & a_{12}^{(2)} & a_{13}^{(3)} & \cdots & a_{1n}^{(n-1)}\\ -\sigma_1 & a_{22}^{(2)} & a_{23}^{(3)} & \cdots & a_{2n}^{(n-1)}\\ & -\sigma_2 & a_{33}^{(3)} & \cdots & a_{3n}^{(n-1)}\\ & & \ddots & \ddots & \vdots\\ & & & -\sigma_{n-1} & a_{nn}^{(n-1)}\end{pmatrix},\text{证毕}.$$

用初等反射矩阵正交相似约化 n 阶实矩阵 $\boldsymbol{A}$ 为上豪斯霍尔德矩阵，由于 $\boldsymbol{U}_1,\boldsymbol{U}_2,\cdots,\boldsymbol{U}_{n-2}$ 都是正交矩阵，所以 $\boldsymbol{A}_1,\boldsymbol{A}_2,\cdots,\boldsymbol{A}_{n-2}$ 都是相似矩阵. 求 $\boldsymbol{A}$ 的特征值问题就转化为求上豪斯霍尔德矩阵 $\boldsymbol{A}_{n-1}$ 的特征值问题. 记 $\boldsymbol{P}=\boldsymbol{U}_{n-2}\boldsymbol{U}_{n-3}\cdots\boldsymbol{U}_2\boldsymbol{U}_1$，则

$$\boldsymbol{A}_{n-1}=\boldsymbol{P}\boldsymbol{A}.$$

设 $\boldsymbol{Y}$ 是 $\boldsymbol{A}_{n-1}=\boldsymbol{P}\boldsymbol{A}$ 的对应于特征值 λ 的特征向量，由 $\boldsymbol{P}^{\mathrm{T}}=\boldsymbol{P}^{-1}$，得 $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{Y}$ 为 $\boldsymbol{A}$ 的对应于特征值 λ 的特征向量.

例 5.5.2 用初等反射矩阵正交相似约化实矩阵 $\boldsymbol{A}$ 为上豪斯霍尔德矩阵，其中

$$\boldsymbol{A}=\begin{pmatrix}-4 & -3 & -7\\ 2 & 3 & 2\\ 4 & 2 & 7\end{pmatrix}.$$

解 **方法 1**（手工计算） 当 $k=1$ 时，取

$$\boldsymbol{A}_1=\boldsymbol{A}=\left(\begin{array}{c|cc}-4 & -3 & -7\\ \hline 2 & 3 & 2\\ 4 & 2 & 7\end{array}\right)=\begin{pmatrix}a_{11} & \boldsymbol{A}_{12}^{(1)}\\ \boldsymbol{A}_{21}^{(1)} & \boldsymbol{A}_{22}^{(1)}\end{pmatrix}.$$

选择初始反射矩阵 $\boldsymbol{P}_1$，使得 $\boldsymbol{P}_1\boldsymbol{A}_{21}^{(1)}=-\sigma_1\boldsymbol{e}_1=\sigma_1(1,0)^{\mathrm{T}}$，则

$$\begin{cases}\sigma_1=\|\boldsymbol{A}_{21}^{(1)}\|_2\operatorname{sign}(a_{21})=\sqrt{20}\approx 4.472\,136,\\ \boldsymbol{\mu}_1=\boldsymbol{A}_{21}^{(1)}+\sigma_1\boldsymbol{e}_1=(2+\sqrt{20},4)^{\mathrm{T}}\approx(6.472\,136,4)^{\mathrm{T}},\\ c_1=\dfrac{1}{2}\|\boldsymbol{\mu}_1\|_2^2=\sigma_1(a_{21}+\sigma_1)=\sqrt{20}(2+\sqrt{20})\approx 28.944\,272,\\ \boldsymbol{P}_1=\boldsymbol{E}-\dfrac{\boldsymbol{\mu}_1\boldsymbol{\mu}_1^{\mathrm{T}}}{c_1}=\begin{pmatrix}-0.447\,214 & -0.894\,427\\ -0.894\,427 & 0.447\,214\end{pmatrix},\end{cases}$$

令 $\boldsymbol{U}_1=\begin{pmatrix}1 & \boldsymbol{O}\\ \boldsymbol{O} & \boldsymbol{P}_1\end{pmatrix}$，则

$$\boldsymbol{A}_2=\boldsymbol{U}_1\boldsymbol{A}_1=\begin{pmatrix}-4 & \boldsymbol{A}_{12}^{(1)}\\ \boldsymbol{P}_1\boldsymbol{A}_{21}^{(1)} & \boldsymbol{P}_1\boldsymbol{A}_{22}^{(1)}\end{pmatrix}=\begin{pmatrix}-4.000\,000 & -3.000\,000 & -7.000\,000\\ -4.472\,136 & -3.130\,495 & -7.155\,418\\ 0.000\,000 & -1.788\,854 & 1.341\,641\end{pmatrix}.$$

方法 2（编写 MATLAB 程序计算） 输入

```
>>A=[-4,-3,-7;2,3,2;4,2,7];k=1;n=3; S1=norm(A(k+1:n,k))
*sign(A(k+1,k)),
u1=A(k+1:n,k)+S1*eye(n-k,1),c1-(norm(u1)^2)/2,
P1=eye(n-k,n-k)-u1*u1'/c1,
```

```
U1 = [eye(k,k),zeros(k,n - k); zeros(n - k, k), P1],A1 = U1 * A,
```

运行后输出结果

```
S1 =                                         c1 =
  4.47213595499958                             28.94427190999916
u1 =                  P1 =
  6.47213595499958        -0.44721359549996  -0.89442719099992
  4.00000000000000        -0.89442719099992   0.44721359549996
U1 =
  1.00000000000000                    0                  0
                 0   -0.44721359549996  -0.89442719099992
                 0   -0.89442719099992   0.44721359549996
A2 =
  -4.00000000000000  -3.00000000000000  -7.00000000000000
  -4.47213595499958  -3.13049516849971  -7.15541752799933
   0.00000000000000  -1.78885438199983   1.34164078649987
```

用 $n-2$ 步正交相似(即豪斯霍尔德)变换将 n 阶矩阵 A 约化成上豪斯霍尔德矩阵,可以用下面编写的 MATLAB 程序完成.

> **用豪斯霍尔德变换将 n 阶矩阵 A 约化成上豪斯霍尔德矩阵的 MATLAB 主程序**
>
> 输入的量:n 阶实对称矩阵 A;
>
> 输出的量:计算全过程中计算次数 k,$S_k=\|A_{22}^{(k)}\|_2\text{sign}(a_{k+1,k}^{(k)})$,$u_k=A_{22}^{(k)}+\sigma_k e_1$,$c_k=\frac{1}{2}\|\mu_k\|_2^2$,$P_k=E-\frac{\mu_k\mu_k^T}{c_k}$,$U_k=\begin{pmatrix}E_k & O\\ O & P_k\end{pmatrix}$,上豪斯霍尔德矩阵 A_k.

```
function [k,Sk,uk,ck,Pk,Uk;Ak] = Householderr1(A)
n = size(A); Ak = A;
for k = 1:n - 2
k,Sk = norm(Ak(k + 1:n,k)) * sign(Ak(k + 1,k)),
uk = Ak(k + 1:n,k) + Sk * eye(n - k,1),
ck = (norm(uk,2)^2)/2,
Pk - eye(n - k,n - k) - uk * uk'/ck,
Uk = [eye(k,k),zeros(k,n - k);zeros(n - k, k),Pk],
A1 - Uk * Ak;Ak - A1,
end
```

例 5.5.3 用初等反射矩阵正交相似约化实矩阵 A 为上豪斯霍尔德矩阵,其中

$$
A=\begin{pmatrix}
12 & -52 & 34 & -12 & 17 & -51\\
-56 & 7 & 2 & 0 & 32 & -17\\
3 & 2 & 5 & 1 & 72 & -63\\
-1 & 0 & 1 & 12 & 21 & -94\\
-32 & -78 & -10.2 & 98 & -72 & 11\\
31 & -41 & -78 & 37 & -19 & 34
\end{pmatrix}.
$$

解 (1) 保存名为 Householdrer1.m 为 M 文件;

(2) 输入 MATLAB 程序

```
>> A=[12 -52 34 -12 17 -51;-56 7 2 0 32 -17;3 2 5 1 72 -63;
-1 0 1 12 21 -94;-32 -78 -10.2 98 -72 11;31 -41 -78 37 -19 34];
[k,Sk,uk,ck,Pk,Uk,Ak]=Householdrer1(A)
```

(3) 运行后屏幕显示结果

```
k =     Sk =          ck =
 1        -71.6310    9.1423e+003
uk =                Pk =
  -127.6310   -0.7818    0.0419   -0.0140   -0.4467    0.4328
     3.0000    0.0419    0.9990    0.0003    0.0105   -0.0102
    -1.0000   -0.0140    0.0003    0.9999   -0.0035    0.0034
   -32.0000   -0.4467    0.0105   -0.0035    0.8880    0.1085
    31.0000    0.4328   -0.0102    0.0034    0.1085    0.8949
Uk =
    1.0000         0         0         0         0         0
         0   -0.7818    0.0419   -0.0140   -0.4467    0.4328
         0    0.0419    0.9990    0.0003    0.0105   -0.0102
         0   -0.0140    0.0003    0.9999   -0.0035    0.0034
         0   -0.4467    0.0105   -0.0035    0.8880    0.1085
         0    0.4328   -0.0102    0.0034    0.1085    0.8949
Ak =
   12.0000  -52.0000   34.0000  -12.0000   17.0000  -51.0000
   71.6310   11.7128  -30.5678  -27.8930    1.6473   21.7643
    0.0000    1.8892    5.7655    1.6556   72.7134  -63.9112
   -0.0000    0.0369    0.7448   11.7815   20.7622  -93.6963
   -0.0000  -76.8184  -18.3655   91.0066  -79.6101   20.7191
    0.0000  -42.1447  -70.0897   43.7749  -11.6277   24.5846
k =     Sk =             ck =
  2      87.6402          7.8464e+003
uk =              Pk =
```

```
    89.5295              -0.0216   -0.0004    0.8765    0.4809
     0.0369              -0.0004    1.0000    0.0004    0.0002
   -76.8184               0.8765    0.0004    0.2479   -0.4126
   -42.1447               0.4809    0.0002   -0.4126    0.7736
Uk =
    1.0000         0         0         0         0         0
         0    1.0000         0         0         0         0
         0         0   -0.0216   -0.0004    0.8765    0.4809
         0         0   -0.0004    1.0000    0.0004    0.0002
         0         0    0.8765    0.0004    0.2479   -0.4126
         0         0    0.4809    0.0002   -0.4126    0.7736
Ak =
   12.0000  -52.0000   34.0000  -12.0000   17.0000  -51.0000
   71.6310   11.7128  -30.5678  -27.8930    1.6473   21.7643
   -0.0000  -87.6402  -49.9272  100.7790  -76.9476   31.4002
   -0.0000   -0.0000    0.7219   11.8223   20.7005  -93.6570
   -0.0000    0.0000   29.4202    5.9564   48.8026  -61.0603
    0.0000    0.0000  -43.8731   -2.8860   58.8230  -20.2818
```

```
k =    Sk =            ck =
 4      -12.2088             195.0398
uk =                  Pk =
   -15.9753              -0.3085    0.9512
    11.6133               0.9512    0.3085
Uk =
    1.0000         0         0         0         0         0
         0    1.0000         0         0         0         0
         0         0    1.0000         0         0         0
         0         0         0    1.0000         0         0
         0         0         0         0   -0.3085    0.9512
         0         0         0         0    0.9512    0.3085
Ak =
   12.0000  -52.0000   34.0000  -12.0000   17.0000  -51.0000
   71.6310   11.7128  -30.5678  -27.8930    1.6473   21.7643
   -0.0000  -87.6402  -49.9272  100.7790  -76.9476   31.4002
    0.0000   -0.0000  -52.8292   -5.8754   21.3902   18.4403
    0.0000    0.0000    0.0000   12.2088   40.2435 -106.8134
    0.0000    0.0000   -0.0000    0.0000   64.7555  -34.0909
```

5.5.3 实对称矩阵的三对角化及其 MATLAB 程序

当 A 是 n 阶实对称矩阵时，由定理 5.25 证明可得出用正交相似变换 $PX = Y$ 约化对称矩阵 A 为三对角矩阵的结果. 三对角矩阵的一般形式为

$$C=\begin{pmatrix} \delta_1 & b_1 & & \\ a_1 & \delta_2 & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & a_{n-1} & \delta_n \end{pmatrix}.$$

推论 5.3 若 A 是 n 阶实对称矩阵，则存在 $n-2$ 个初等反射矩阵 $U_1, U_2, \cdots, U_{n-2}$，使得 $A_{n-1} = U_{n-2}\cdots U_2U_1AU_1U_2\cdots U_{n-2}$为三对角矩阵，即

$$A_{n-1}=\begin{pmatrix} a_{11} & -\sigma_1 & & & \\ -\sigma_1 & a_{22}^{(2)} & -\sigma_2 & & \\ & -\sigma_2 & a_{33}^{(3)} & \ddots & \\ & & \ddots & \ddots & -\sigma_{n-1} \\ & & & -\sigma_{n-1} & a_{nn}^{(n-1)} \end{pmatrix}.$$

用 $n-2$ 步正交相似变换将 n 阶实对称矩阵 A 约化成三对角矩阵，可以用下面的 MATLAB 程序完成.

将 n 阶实对称矩阵 A 约化成三对角矩阵的 MATLAB 主程序
输入的量：n 阶实对称矩阵 A；
输出的量：三对角矩阵 A.

```
function T = house(A)
[n,n] = size(A);
for k = 1:n - 2
    s = norm(A(k + 1:n,k));
    if (A(k + 1,k) < 0)
        s = - s;
    end
    r = sqrt(2 * s * (A(k + 1,k) + s)); U(1:k) = zeros(1,k);
    U(k + 1) = (A(k + 1,k) + s)/r;
    U(k + 2:n) = A(k + 2:n,k)'/r;V(1:k) = zeros(1,k);
    V(k + 1:n) = A(k + 1:n,k + 1:n) * U(k + 1:n)';C = U(k + 1:n) * V(k + 1:n)';
    P(1:k) = zeros(1,k); P(k + 1:n) = V(k + 1:n) - C * U(k + 1:n);
    A(k + 2:n,k) = zeros(n - k - 1,1);
    A(k,k + 2:n) = zeros(1,n - k - 1); A(k + 1,k) = - s; A(k,k + 1) - - s;
  A(k + 1:n,k + 1:n) = A(k + 1:n,k + 1:n) - 2 * U(k + 1:n)' * P(k + 1:n) - 2
```

```
*P(k+1:n)'*U(k+1:n);
     end
     T=A;
```

例 5.5.4 用初等反射矩阵正交相似约化实对称矩阵 A 为三对角矩阵，其中

$$A=\begin{pmatrix} 12 & -56 & 3 & -14 & -90 & -4 \\ -56 & 71 & 23 & 61 & -9 & -21 \\ 3 & 23 & 53 & 12 & -72 & 51 \\ -14 & 61 & 12 & 73 & 23 & 21 \\ -90 & -9 & -72 & 23 & -34 & -61 \\ -41 & -21 & 51 & 21 & -61 & -52 \end{pmatrix}.$$

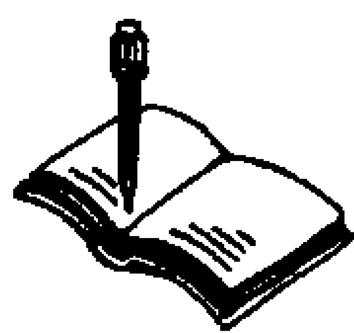
解 (1) 保存名为 house.m 为 M 文件；

(2) 输入 MATLAB 程序

```
>> A=[12 -56 3 -14 -90 -4;-56 71 23 61 -9 -21;3 23 53 12 -72
51;-14 61 12 73 23 21;-90 -9 -72 23 -34 -61;-41 -21 51 21 -61 -52];
T=house(A)
```

(3) 运行后屏幕显示结果

```
T =
     12.0000   114.5513          0          0          0          0
    114.5513   -43.2395  -108.2763          0          0          0
           0  -108.2763    49.7411   -22.7766          0          0
           0          0    22.7766    40.2476   -89.1355          0
           0          0          0   -89.1355    44.9606    39.3090
           0          0          0          0    39.3090    19.2902
```



习 题 5.5

1. 用初等反射矩阵正交相似约化实对称矩阵 A 为三对角矩阵，其中

$$A=\begin{pmatrix} -92 & -5 & 3 & -14 & -90 & -41 \\ -5 & 71 & 23 & 61 & -9 & -21 \\ 3 & 23 & 53 & 12 & -72 & 51 \\ -14 & 61 & 12 & 73 & 23 & 21 \\ -90 & -9 & -72 & 23 & -34 & -61 \\ -41 & -21 & 51 & 21 & -61 & -52 \end{pmatrix}.$$

2. 用初等反射矩阵正交相似约化实矩阵 A 为上豪斯霍尔德矩阵，其中

$$A=\begin{pmatrix}67 & -12 & 34 & -12 & 17 & -51\\ -56 & 7 & 2 & 0 & 32 & -17\\ 3 & 2 & 5 & 1 & 72 & -63\\ -1 & 0 & 1 & 12 & 21 & -94\\ -32 & -78 & -10.2 & 98 & -72 & 11\\ 31 & -41 & -78 & 37 & -19 & 34\end{pmatrix}.$$

5.6 QR 方法及其 MATLAB 程序

QR 方法是一种变换方法，它是计算中小型矩阵的全部特征值问题的最有效的方法之一. 这种方法主要计算上豪斯霍尔德矩阵和对称三对角矩阵的全部特征值问题，并且收敛速度快，算法稳定. 对于一般的矩阵 $\boldsymbol{A}$，首先用正交相似变换(即豪斯霍尔德方法)约化为上豪斯霍尔德矩阵或对称三对角矩阵，然后用 QR 方法计算该上豪斯霍尔德矩阵或对称三对角矩阵的全部特征值问题.

5.6.1 QR 方法及其收敛性

定义 5.9 如果 n 阶方阵 $\boldsymbol{A}$ 可以分解为 $\boldsymbol{A}=\boldsymbol{QR}$ 的形式，其中 $\boldsymbol{Q}$ 为 n 阶正交矩阵，$\boldsymbol{R}$ 为 n 阶上三角形矩阵，则称 $\boldsymbol{A}$ 可 QR **分解**，也称 $\boldsymbol{A}$ **可正交三角分解**.

定理 5.26 如果 n 阶实矩阵 $\boldsymbol{A}=(a_{ij})_{n\times n}$ 是非奇异的，则存在 $\boldsymbol{A}$ 的正交三角分解 $\boldsymbol{A}=\boldsymbol{QR}$，其中 $\boldsymbol{Q}$ 为 n 阶正交矩阵，$\boldsymbol{R}$ 为 n 阶非奇异上三角形矩阵. 如果限定 $\boldsymbol{R}$ 的主对角线上的元全为正数，则此种分解唯一.

证明 下面用数学归纳法证明.

(1) 取 $\boldsymbol{A}_1=\boldsymbol{A}=(\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_n)$，其中 $\boldsymbol{a}_j$ 表示 $\boldsymbol{A}$ 的第 $j(j=1,2,\cdots,n)$ 列向量. 由 $\boldsymbol{A}$ 是非奇异矩阵，知 $\boldsymbol{a}_1\neq\boldsymbol{0}$. 根据推论 5.2 知，对于 $\boldsymbol{a}_1$ 可选择初始反射矩阵 $\boldsymbol{P}_1=\boldsymbol{E}-\dfrac{\boldsymbol{\mu}_1\boldsymbol{\mu}_1^{\mathrm{T}}}{c_1}$，使得 $\boldsymbol{P}_1\boldsymbol{a}_1=-\sigma_1\boldsymbol{e}_1$，其中 $\sigma_1=\|\boldsymbol{a}_1\|_2\operatorname{sign}(a_{11})$，$\boldsymbol{\mu}_1=\boldsymbol{a}_1+\sigma_1\boldsymbol{e}_1$，$c_1=\dfrac{1}{2}\|\boldsymbol{\mu}_1\|_2^2=\sigma_1(a_{11}+\sigma_1)$.

令 $\boldsymbol{U}_1=\begin{pmatrix}\boldsymbol{E}_1 & \boldsymbol{O}\\ \boldsymbol{O} & \boldsymbol{P}_1\end{pmatrix}$，则

$$\boldsymbol{A}_2=\boldsymbol{U}_1\boldsymbol{A}_1=\left(\begin{array}{c|c|c|c}\sigma_1 & a_{12}^{(2)} & \cdots & a_{1n}^{(2)}\\ \hline 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)}\\ \vdots & \vdots & & \vdots\\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)}\end{array}\right)=\begin{pmatrix}\sigma_1 & \boldsymbol{A}_{12}^{(2)} & \boldsymbol{A}_{13}^{(2)}\\ \boldsymbol{O} & \boldsymbol{A}_{22}^{(2)} & \boldsymbol{A}_{23}^{(2)}\end{pmatrix}.$$

(2) 由 $\boldsymbol{A}$ 是非奇异矩阵，知 $\boldsymbol{A}_{22}^{(2)}=(a_{22}^{(2)},a_{32}^{(2)},\cdots,a_{n2}^{(2)})^{\mathrm{T}}\neq\boldsymbol{O}$. 根据推论 5.2

知，对于 $A_{22}^{(2)}$ 可选择初始反射矩阵 $P_2 = E - \frac{\mu_2 \mu_2^T}{c_2}$，使得 $P_2 A_{22}^{(2)} = -\sigma_2 e_1$，其中 $\sigma_2 = \|A_{22}^{(2)}\|_2 \mathrm{sign}(a_{22}^{(2)})$，$\mu_2 = A_{22}^{(2)} + \sigma_2 e_1$，$c_2 = \frac{1}{2}\|\mu_2\|_2^2 = \sigma_2(a_{22}^{(2)} + \sigma_2)$.

令 $U_2 = \begin{pmatrix} E_2 & O \\ O & P_2 \end{pmatrix}$，则

$$A_3 = U_2 A_2 = \left(\begin{array}{cc|c|cc} -\sigma_1 & a_{12}^{(2)} & a_{13}^{(3)} & \cdots & a_{1n}^{(3)} \\ 0 & -\sigma_2 & a_{23}^{(3)} & \cdots & a_{2n}^{(3)} \\ \hline 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} \end{array}\right) = \begin{pmatrix} A_{11}^{(3)} & A_{12}^{(3)} & A_{13}^{(3)} \\ O & A_{22}^{(3)} & A_{23}^{(3)} \end{pmatrix}.$$

(3) 如此继续下去，经过对 A 进行 $k-1$ 步正交相似约化，A_k 具有如下形式

$$A_k = U_{k-1} A_{k-1} = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} & A_{13}^{(k)} \\ O & A_{22}^{(k)} & A_{23}^{(k)} \end{pmatrix},$$

其中 $A_{11}^{(k)} \in \mathbf{R}^{k \times k}$，$A_{22}^{(k)} \in \mathbf{R}^{n-k}$，$A_{23}^{(k)} \in \mathbf{R}^{(n-k)\times(n-k)}$.

由 A 是非奇异矩阵，知 $A_{22}^{(k)} = (a_{kk}^{(k)}, a_{k+1,k}^{(k)}, \cdots, a_{nk}^{(k)})^T \neq O$. 根据推论 5.2 知，对于 $A_{22}^{(k)}$ 选择初始反射矩阵 P_k，使得 $P_k A_{22}^{(k)} = -\sigma_k e_1$，其中

$$\begin{cases} \sigma_k = \|A_{22}^{(k)}\|_2 \mathrm{sign}(a_{kk}^{(k)}), \\ \mu_k = A_{22}^{(k)} + \sigma_k e_1, \\ c_k = \frac{1}{2}\|\mu_k\|_2^2 = \sigma_k(a_{kk}^{(k)} + \sigma_k), \\ P_k = E - \frac{\mu_k \mu_k^T}{c_k}. \end{cases}$$

令 $U_k = \begin{pmatrix} E_k & O \\ O & P_k \end{pmatrix}$，则

$$A_{k+1} = U_k A_k = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} & A_{13}^{(k)} \\ O & P_k A_{22}^{(k)} & P_k A_{23}^{(k)} \end{pmatrix}, \tag{5.56}$$

其中 $A_{11}^{(k)}$ 是上三角形矩阵.

由数学归纳法可以证明，当 $k = n-1$ 时，(5.56)式为上三角形矩阵

$$A_n = U_{n-1} A_{n-1} = U_{n-1} \cdots U_2 U_1 A,$$

即

$$A_n=\begin{pmatrix} -\sigma_1 & a_{12}^{(2)} & a_{13}^{(2)} & \cdots & a_{1,n-1}^{(n-2)} & a_{1n}^{(n-1)} \\ 0 & -\sigma_2 & a_{23}^{(3)} & \cdots & a_{2,n-1}^{(n-2)} & a_{2n}^{(n-1)} \\ 0 & 0 & -\sigma_3 & & \ddots & \vdots \\ \vdots & \vdots & \ddots & & \ddots & a_{n-1,n}^{(n-1)} \\ 0 & 0 & \cdots & & 0 & -\sigma_n \end{pmatrix}.$$

设 $\boldsymbol{A}_n=\boldsymbol{R},\boldsymbol{P}=\boldsymbol{U}_{n-1}\boldsymbol{U}_{n-2}\cdots\boldsymbol{U}_2\boldsymbol{U}_1,\boldsymbol{P}^{-1}=\boldsymbol{Q}$. 由于 $\boldsymbol{U}_1,\boldsymbol{U}_2,\cdots,\boldsymbol{U}_{n-2}$都是正交矩阵,则 $\boldsymbol{P}$ 和 $\boldsymbol{Q}$ 仍是正交矩阵. 由 $\boldsymbol{A}_n=\boldsymbol{PA}$,得 $\boldsymbol{A}=\boldsymbol{QR}$. 即利用 $n-1$ 个初等反射矩阵,实现了 $\boldsymbol{A}$ 的 QR 分解.

现在证唯一性. 设有

$$\boldsymbol{A}=\boldsymbol{Q}_1\boldsymbol{R}_1=\boldsymbol{Q}_2\boldsymbol{R}_2.$$

其中 $\boldsymbol{Q}_1,\boldsymbol{Q}_2$ 为 n 阶正交矩阵,$\boldsymbol{R}_1,\boldsymbol{R}_2$ 为 n 阶上三角形矩阵,于是

$$\boldsymbol{Q}_2^{\mathrm{T}}\boldsymbol{Q}_1=\boldsymbol{R}_2\boldsymbol{R}_1^{-1}. \tag{5.57}$$

由(5.57)式知,上三角形矩阵 $\boldsymbol{R}_2\boldsymbol{R}_1^{-1}$ 为正交矩阵,故为对角矩阵,即

$$\boldsymbol{R}_2\boldsymbol{R}_1^{-1}=\boldsymbol{D}=\mathrm{diag}(d_1,d_2,\cdots,d_n),且\ \boldsymbol{D}^2=\boldsymbol{E}.$$

因为 $\boldsymbol{R}_2,\boldsymbol{R}_1$ 的对角元都为正数,故 $d_i>0\ (i=1,2,\cdots,n)$,即 $\boldsymbol{D}=\boldsymbol{E}$. 于是 $\boldsymbol{R}_2=\boldsymbol{R}_1$. 由(5.57)式知,$\boldsymbol{Q}_2=\boldsymbol{Q}_1$. 证毕.

定理 5.27(基本 QR 方法) 如果 n 阶实矩阵 $\boldsymbol{A}$ 是非奇异的,QR 算法如下:

$$\begin{cases}\boldsymbol{A}_1=\boldsymbol{A},\\ \boldsymbol{A}_k=\boldsymbol{Q}_k\boldsymbol{R}_k, \quad (k=1,2,\cdots,n),\\ \boldsymbol{A}_{k+1}=\boldsymbol{R}_k\boldsymbol{Q}_k\end{cases} \tag{5.58}$$

其中 $\boldsymbol{Q}_k$ 为 n 阶正交矩阵,$\boldsymbol{R}_k$ 为 n 阶非奇异上三角形矩阵. 记 $\boldsymbol{Q}=\boldsymbol{Q}_1\boldsymbol{Q}_2\cdots\boldsymbol{Q}_k,\boldsymbol{R}=\boldsymbol{R}_k\cdots\boldsymbol{R}_2\boldsymbol{R}_1$, 则有

(1) $\boldsymbol{A}_{k+1}=\boldsymbol{Q}_k^{\mathrm{T}}\boldsymbol{A}_k\boldsymbol{Q}_k(k=1,2,\cdots,n)$;

(2) $\boldsymbol{A}_{k+1}=(\boldsymbol{Q}_1\boldsymbol{Q}_2\cdots\boldsymbol{Q}_k)^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Q}_1\boldsymbol{Q}_2\cdots\boldsymbol{Q}_k$,即 $A_{k+1}=\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Q}$;

(3) $\boldsymbol{A}^k$ 的 QR 分解式为 $\boldsymbol{A}^k=\boldsymbol{QR}$.

定理 5.28(基本 QR 方法的收敛性) 如果 n 阶实矩阵 $\boldsymbol{A}=(a_{ij})_{n\times n}$是非奇异的,且满足如下条件:

(1) $\boldsymbol{A}$ 的全部特征值满足:$|\lambda_1|>|\lambda_2|>\cdots>|\lambda_n|>0$;

(2) $\boldsymbol{A}$ 的标准形 $\boldsymbol{A}=\boldsymbol{XDX}^{-1}$,其中 $\boldsymbol{D}=\mathrm{diag}(\lambda_1,\lambda_2,\cdots,\lambda_n)$,且 $\boldsymbol{X}^{-1}$有三角分解 $\boldsymbol{X}^{-1}=\boldsymbol{LU}$,其中 $\boldsymbol{L}$ 为 n 阶单位下三角形矩阵,$\boldsymbol{U}$ 为 n 阶上三角形矩阵.

则由 QR 算法产生的迭代序列$\{\boldsymbol{A}_k\}$本质上收敛于上三角形矩阵,即

$$A_k \xrightarrow{\text{本质上}} R = \begin{pmatrix} \lambda_1 & * & \cdots & * \\ & \lambda_2 & \cdots & * \\ & & \ddots & \vdots \\ & & & \lambda_n \end{pmatrix} \quad (\text{当 } k\to\infty \text{ 时}),$$

或者(1) $\lim\limits_{k\to\infty} a_{ii}^{(k)} \to \lambda_i$;

(2) $\lim\limits_{k\to\infty} a_{ij}^{(k)} = \begin{cases} 0, & \text{当 } i>j \text{ 时}, \\ \text{不一定存在}, & \text{当 } i<j \text{ 时}. \end{cases}$

定理 5.29 如果 n 阶非奇异矩阵 $A=(a_{ij})_{n\times n}$ 是对称矩阵，则由 QR 算法产生的迭代序列 $\{A_k\}$ 收敛于对角矩阵 $D=\mathrm{diag}(\lambda_1,\lambda_2,\cdots,\lambda_n)$.

5.6.2 QR 方法的 MATLAB 程序

QR 方法也称作正交三角分解(Orthogonal-triangular decomposition). 在MATLAB函数库中有编好的程序，我们只要直接调用这些程序即可，调用方法见表 5-4.

表 5-4

命令	功能
[Q,R] = qr (A)	(1) 如果输入的 n 阶矩阵 A 是非奇异的，运行后输出 n 阶非奇异的上三角形矩阵 R 和 n 阶正交矩阵 Q，使得 $A=QR$； (2) 如果输入的 n 阶矩阵 A 是奇异的，运行后输出 n 阶奇异的上三角形矩阵 R 和 n 阶正交矩阵 Q，使得 $A=QR$； (3) 如果输入的矩阵 A 是 $m\times n$ 阶，且 $m>n$，则运行后输出 $m\times n$阶上三角形矩阵 R(其中 R 的 $m-n$ 行和这行以后的元全为零)和 m 阶正交矩阵 Q，使得 $A=QR$，其中实际上仅仅是 Q 的前 n 列与 R 计算. 请参考例 5.6.1 和例 5.6.2.
[Q,R,E] = qr (A)	(1) 如果输入的 n 阶矩阵 A 是非奇异的，运行后输出 n 阶置换矩阵 E，n 阶非奇异的上三角形矩阵 R 和 n 阶正交矩阵 Q，使得 $AE=QR$. (2) 如果输入的 n 阶矩阵 A 是奇异的，运行后输出 n 阶置换矩阵 E，n 阶奇异的上三角形矩阵 R 和 n 阶正交矩阵 Q，使得 $AE=QR$； (3) 如果输入的矩阵 A 是 $m\times n$ 阶，且 $m>n$，则运行后输出 $m\times n$阶上三角形矩阵 R(其中 R 的 $m-n$ 行和这行以后的元全为零)和 m 阶正交矩阵 Q，使得 $AE=QR$，其中实际上仅仅是 Q 的前 n 列与 R 计算. 说明：输出的 n 阶列置换矩阵 E 使得上三角形矩阵 R 的主对角线上的元的绝对值按列从大到小排列. 请参考例 5.6.1 和例 5.6.2.

续表

命令	功能
[Q,R] = qr (A,0)	[Q,R] = QR(A,0)产生“economy size”分解. (1) 如果输入的是 n 阶方阵 $\boldsymbol{A}$,则运行后输出的结果与[Q,R] = qr(A)相同; (2) 如果输入的矩阵 $\boldsymbol{A}$ 是 $m\times n$ 阶,且 $m>n$,则运行后输出 n 阶上三角形矩阵 $\boldsymbol{R}$ 和 $m\times n$ 阶矩阵 $\boldsymbol{Q}$(其中 $\boldsymbol{Q}$ 的列向量两两正交),使得 $\boldsymbol{Q}$ 的前 n 列与 $\boldsymbol{R}$ 的积等于 $\boldsymbol{A}$. 请参考例 5.6.2.
[Q,R,E] = qr (A,0)	[Q,R,E] = QR(A,0) 产生“economy size”分解,使得 $\boldsymbol{QR}=\boldsymbol{A}(:,\boldsymbol{E})$. 其中 $\boldsymbol{E}$ 是置换向量,$\boldsymbol{E}$ 中的元表示元 1 所在的行数. 例 $\boldsymbol{E}=(1\quad 3\quad 2)$表示置换矩阵为 $\boldsymbol{E}_1=\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}$ 其他功能与上类似. 请参考例 5.6.2.

例 5.6.1 用[Q,R] = qr(A)和[Q,R,E] = qr (A)将矩阵 $\boldsymbol{A}$ 进行正交三角分解,并且比较差异.

(1) $\boldsymbol{A}=\begin{pmatrix}12&-52&34&-12&17&-51\\-56&7&2&0&32&-17\\3&2&5&1&72&-63\\-1&0&1&12&21&-94\\-32&-78&-51/5&98&-72&11\\31&-41&-78&37&-19&34\end{pmatrix}$;

(2) $\boldsymbol{A}=\begin{pmatrix}1&2&3\\1&2&3\\3&4&5\end{pmatrix}$.

解 (1) 输入 MATLAB 程序

```
>> A=[12 -52 34 -12 17 -51;-56 7 2 0 32 -17;3 2
5 1 72 -63; -1 0 1 12 21 -94;-32 -78 -51/5 98 -72
11;31 -41 -78 37 -19 34];
dA=det(A),
[Q,R] = qr(A),[Q1,R1,E] = qr (A)
```

运行后屏幕显示结果

```
Q =
```

```
   -0.1652   -0.5119   -0.6275    0.5419    0.0963   -0.1181
    0.7710    0.0905    0.2397    0.5492    0.1788   -0.0789
   -0.0413    0.0183   -0.0708   -0.1448    0.8928    0.4183
    0.0138    0.0004   -0.0083   -0.2650    0.3734   -0.8889
    0.4406   -0.7480    0.0514   -0.4731   -0.0884    0.1099
   -0.4268   -0.4122    0.7356    0.2995    0.1201   -0.0524
R =
   -72.6292   -2.9602   24.5301   29.4923   -4.4335  -13.0388
          0  102.5341   22.6482  -82.3892   57.2090    1.1353
          0         0  -79.1162   39.6136  -25.9425   58.7372
          0         0         0  -45.1109   39.1738    2.0332
          0         0         0         0   83.5636  -96.1856
          0         0         0         0         0   63.9909
Q1 =
   -0.3915    0.0638   -0.1953    0.3892    0.8069    0.0450
   -0.1305   -0.0167    0.7920   -0.4739    0.3601   -0.0334
   -0.4836   -0.0715   -0.1164   -0.1164   -0.1540   -0.8427
   -0.7215   -0.2064   -0.1251   -0.2487   -0.2738    0.5332
    0.0844   -0.9202    0.1887    0.3297    0.0028   -0.0425
    0.2610   -0.3181   -0.5194   -0.6632    0.3472   -0.0229
R1 =
   130.2766   13.4867    7.2691  -37.9285    1.1898  -71.8394
          0 -105.2621   21.2806   35.7685   81.2358   63.3668
          0         0  -69.0601   32.8274   22.0444    7.3019
          0         0         0   59.8201  -22.3106  -33.2933
          0         0         0         0  -54.1998    1.6040
          0         0         0         0         0  -46.2862
E =                                       dA =
     0     0     1     0     0     0         1.4212e+011
     0     0     0     0     1     0
     0     0     0     1     0     0
     0     1     0     0     0     0
     0     0     0     0     0     1
     1     0     0     0     0     0
```

因为 $\boldsymbol{A}$ 的行列式 $\det \boldsymbol{A} = 1.4212\mathrm{e}+011 \neq 0$，所以 $\boldsymbol{A}$ 是非奇异矩阵，上三角形矩阵 $\boldsymbol{R}$ 和 $\boldsymbol{R}_1$ 都是非奇异矩阵，且不相同. 输入程序

```
>> Q1' - inv(Q1),Q' - inv(Q)
```

由运行后输出的结果可知 $\boldsymbol{Q}^{\mathrm{T}} = \boldsymbol{Q}^{-1}$，且 $\boldsymbol{Q}_1^{\mathrm{T}} = \boldsymbol{Q}_1^{-1}$，故 $\boldsymbol{Q}$ 和 $\boldsymbol{Q}_1$ 都是正交矩阵，但

是不相等.

再输入程序

```
>> abs(diag(R1))
```

运行后输出结果经整理得

$$(130.2766 \quad 105.2621 \quad 69.0601 \quad 59.8201 \quad 54.1998 \quad 46.2862)^T$$

说明 输出的6阶列置换矩阵 $\boldsymbol{E}$ 使得上三角形矩阵 $\boldsymbol{R}_1$ 的主对角线上元的绝对值按列从大到小排列.

(2) 输入 MATLAB 程序

```
>> A=[1 2 3;1 2 3;3 4 5]; [Q,R] = qr(A),
   [Q1,R1,E] = qr (A), dA=det(A)
```

运行后屏幕显示结果

```
Q =                                     R =
     -0.3015    0.6396   -0.7071     -3.3166   -4.8242   -6.3317
     -0.3015   -0.6396    0.7071           0   -0.8528   -1.7056
     -0.9045    0.4264   -0.0000           0         0    0.0000
Q1 =                                    R1 =
     -0.4575   -0.5392   -0.7071     -6.5574   -3.2025   -4.8800
     -0.4575   -0.5392    0.7071           0    0.8627    0.4313
     -0.7625    0.6470    0.0000           0         0    0.0000
E =                               dA =
       0    1    0                     0
       0    0    1
       1    0    0
```

因为 $\boldsymbol{A}$ 的行列式 $\det \boldsymbol{A}=0$,所以 $\boldsymbol{A}$ 是奇异矩阵,上三角形矩阵 $\boldsymbol{R}$ 和 $\boldsymbol{R}_1$ 都是奇异矩阵,且不相同. 由运行程序 Q1'-inv(Q1),Q'-inv(Q)后输出的结果,可知 $\boldsymbol{Q}_1^T=\boldsymbol{Q}_1^{-1}$ 且 $\boldsymbol{Q}^T=\boldsymbol{Q}^{-1}$,故 $\boldsymbol{Q}_1$ 和 $\boldsymbol{Q}$ 都是正交矩阵,但是不相等. 运行程序

```
>> abs(diag(R1))
```

运行后输出结果经整理得

$$(3.3166 \quad 0.8528 \quad 0.0000)^T.$$

说明 3阶列置换矩阵 $\boldsymbol{E}$ 使得上三角形矩阵 $\boldsymbol{R}_1$ 的主对角线上的元的绝对值按列从大到小排列. 用程序

```
A1=A*E-Q1*R1,     A2=Q*R-A
```

可以验证: $\boldsymbol{AE}=\boldsymbol{Q}_1\boldsymbol{R}_1$ 和 $\boldsymbol{A}=\boldsymbol{QR}$.

例 5.6.2 用[Q,R] = qr(A),[Q1,R1,E1] = qr (A),[Q2,R2] = qr(A,0),[Q3,R3,E3] = qr (A,0),将矩阵 $\boldsymbol{A}$ 进行正交三角分解,并且比较差异.

(1) $A=\begin{pmatrix}1&2&3\\11&2&3\\3&4&5\\7&8&9\\12&3&8\end{pmatrix}$; (2) $A=\begin{pmatrix}2&1&5&0&17\\-1&3&2&9&-7\\21&3&8&-6&11\end{pmatrix}$; (3) $A=\begin{pmatrix}7&2&3\\1&-5&3\\3&4&5\end{pmatrix}$.

解 (1) 输入 MATLAB 程序

```
>> A=[1  2  3;11  2  3;3  4  5;7  8  9;12  3  8];
[Q,R] = qr(A), [Q1,R1,E1] = qr (A),
[Q2,R2] = qr(A,0), [Q3,R3,E3] = qr (A,0)
```

运行后屏幕显示结果

```
Q =
   -0.0556    0.2355    0.2776   -0.3787   -0.8491
    0.6111    0.3442    0.6474   -0.2560   -0.1530
   -0.1667    0.4131    0.1136   -0.7391    0.4923
   -0.3889    0.7682   -0.2145    0.4597   -0.0366
   -0.6667   -0.2555    0.6670    0.1828    0.1093
R =
   -18.0000    7.1111   -11.6667
          0    6.8141     6.6094
          0         0     2.8644
          0         0          0
          0         0          0
Q1 =
   -0.0556    0.3265    0.1610   -0.3405   -0.8651
   -0.6111    0.5733   -0.4571   -0.2489   -0.1642
   -0.1667    0.4242   -0.0601   -0.7603    0.4589
   -0.3889    0.6196   -0.5023    0.4609   -0.0161
   -0.6667    0.0308    0.7136    0.1778    0.1173
R1 =
   -18.0000   -11.6667   -7.1111
          0     7.2034    6.2522
          0          0   -2.7096
          0          0         0
          0          0         0
E1 =
     1     0     0
     0     0     1
     0     1     0
```

```
Q2 =
   -0.0556    0.2355    0.2776
   -0.6111   -0.3442   -0.6474
   -0.1667    0.4131    0.1136
   -0.3889    0.7682   -0.2145
   -0.6667   -0.2555    0.6670
R2 =
   -18.0000   -7.1111   -11.6667
          0    6.8141     6.6094
          0         0     2.8644
Q3 =
   -0.0556    0.3265    0.1610
   -0.6111   -0.5733   -0.4571
   -0.1667    0.4242   -0.0601
   -0.3889    0.6196   -0.5023
   -0.6667    0.0308    0.7136
R3 =
   -18.0000   -11.6667   -7.1111
          0     7.2034    6.2522
          0          0    2.7096
E3 =
     1     3     2
```

因为输入的矩阵 $\boldsymbol{A}$ 是 5×3 阶,且 $m=5>n=3$,由输出的结果可见:

① 用程序[Q,R] = qr(A)和[Q1,R1,E1] = qr (A)输出的结果:矩阵 $\boldsymbol{R}$ 和 $\boldsymbol{R}_1$ 都是第 4 行和第 5 行都为零元的 5×3 阶上三角形矩阵,但是不相等. 5 阶正交矩阵 $\boldsymbol{Q}$ 和 $\boldsymbol{Q}_1$ 也不相等;

② 用程序[Q3,R3,E3] = qr (A,0)和[Q2,R2] = qr(A,0)输出的结果:矩阵 $\boldsymbol{R}$ 和 $\boldsymbol{R}_3$ 都是 3 阶上三角形方阵,但不相等. 5×3 阶矩阵 $\boldsymbol{Q}_2$ 和 $\boldsymbol{Q}_3$ 也不相等,其中它们的列向量分别两两正交,使得 $\boldsymbol{Q}_2$ 和 $\boldsymbol{Q}_3$ 的前 3 列与 $\boldsymbol{R}$ 的积等于 $\boldsymbol{A}$;

③ 用程序[Q3,R3,E3] = qr (A,0)和[Q1,R1,E1] = qr (A)输出的置换矩阵不同,$\boldsymbol{E}_3$ 是三维行向量,而 $\boldsymbol{E}_1$ 是 3 阶方阵.

(2) 输入 MATLAB 程序

```
>> A=[2  1  5  0  17;-1  3  2  9  -7;21  3  8  -6  11];
[Q,R] = qr(A),[Q1,R1,E1] = qr (A),
[Q2,R2] = qr(A,0),[Q3,R3,E3] = qr (A,0)
```

运行后屏幕显示结果

```
Q =
```

```
    -0.0947   -0.2241   -0.9700
     0.0474   -0.9742    0.2204
    -0.9944   -0.0251    0.1029
R =
  -21.1187   -2.9358   -8.3338    6.3924  -12.8796
         0   -3.2220   -3.2693   -8.6179    2.7349
         0         0   -3.5859    1.3668  -16.9008
Q1 =
   -0.7935    0.4784   -0.3761
    0.3267   -0.1865   -0.9265
   -0.5134   -0.8581   -0.0083
R1 =
   -21.4243  -12.6959    6.0212   -7.4215   -1.3536
          0  -16.8765    3.4697   -4.8457   -2.6555
          0         0   -8.2889   -3.8001   -3.1806
E1 =
     0     1     0     0     0
     0     0     0     0     1
     0     0     0     1     0
     0     0     1     0     0
     1     0     0     0     0
Q2 =
   -0.0947   -0.2241   -0.9700
    0.0474   -0.9742    0.2204
   -0.9944   -0.0251    0.1029
R2 =
   -21.1187   -2.9358   -8.3338    6.3924  -12.8796
          0   -3.2220   -3.2693   -8.6179    2.7349
          0         0   -3.5859    1.3668  -16.9008
Q3 =
   -0.7935    0.4784   -0.3761
    0.3267   -0.1865   -0.9265
   -0.5134   -0.8581   -0.0083
R3 =
   -21.4243  -12.6959    6.0212   -7.4215   -1.3536
          0  -16.8765    3.4697   -4.8457   -2.6555
          0         0   -8.2889   -3.8001   -3.1806
E3 =
```

```
     5     1     4     3     2
```

因为输入的矩阵 $\boldsymbol{A}$ 是 3×5 阶矩阵，且 $m=3>n=5$，由输出的结果可见：

① 用[Q,R] = qr(A)和[Q1,R1,E1] = qr (A)程序，输出的结果：上三角形矩阵 $\boldsymbol{R}$ 和 $\boldsymbol{R}_1$ 都是 3×5 阶矩阵，但不相等. 3 阶正交矩阵 $\boldsymbol{Q}$ 和 $\boldsymbol{Q}_1$ 也不相等；

② 用[Q,R] = qr(A)，[Q2,R2] = qr(A,0)输出的 3 阶正交矩阵 $\boldsymbol{Q}_3$ 和 $\boldsymbol{Q}_1$ 相同，且三角形矩阵 $\boldsymbol{R}_3$ 和 3×5 阶矩阵 $\boldsymbol{R}_1$ 不相等；

③ 运行程序[Q3,R3,E3] = qr (A,0)和[Q1,R1,E1] = qr (A)后输出的置换矩阵不同，$\boldsymbol{E}_3$ 是 5 维行向量，而 $\boldsymbol{E}_1$ 是 5 阶方阵. 输出的 3 阶正交矩阵 $\boldsymbol{Q}_3$ 和 $\boldsymbol{Q}_1$ 相同，且三角形矩阵 $\boldsymbol{R}_3$ 和 $\boldsymbol{R}_1$ 都是 3×5 阶矩阵，且不相等.

(3) 输入 MATLAB 程序

```
>> A=[7  2  3;1  -5  3;3  4  5];[Q,R] = qr(A),
[Q1,R1,E1] = qr (A),[Q2,R2] = qr(A,0),[Q3,R3,E3] = qr (A,0)
```

运行后屏幕显示结果略. 请读者自己总结.

例 5.6.3 用 QR 方法解下面的线性方程组，然后再用其他方法验证解的正确性.

(1) $\begin{cases}5x+4y+5z=1,\\7x+8y+9z=2,\\12x+3y+8z=3;\end{cases}$ (2) $\begin{cases}3x+4y+5z=1,\\6x+8y+10z=2,\\12x+3y+8z=3;\end{cases}$ (3) $\begin{cases}3x+4y+5z=4,\\7x+8y+9z=2,\\12x+3y+8z=3.\end{cases}$

解 (1) **解法 1** 输入 MATLAB 程序

```
>> A=[5  4  5;7  8  9;12  3  8];b=[1  2  3]';
[Q R] = qr(A);x = R\(R'\(A'*b)), r = b - A*x;
e = R\(R'\(A'*r)); x1 = x + e
```

运行后屏幕将显示结果

```
x =                            x1 =
    0.5000                         -0.5000
    1.0000                         -1.0000
    1.5000                          1.5000
```

解法 2 输入 MATLAB 程序

```
>> x1 = A\b
```

运行后屏幕显示与解法 1 结果相同.

(2) **解法 1** 输入 MATLAB 程序

```
>> A=[3  4  5;6  8  10;12  3  8];b=[1  2  3]';
[Q R] = qr(A);x1 = R\(R'\(A'*b)),r = b - A*x1;
e = R\(R'\(A'*r)); x = x1 + e
```

运行后屏幕将显示结果

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 6.460547e-017.
x1 =                    x =
   1.0e+016 *             1.0e+017 *
    -0.7362                 0.8687
    -1.5589                 1.8395
     1.6888                -1.9928
```

解法 2 输入 MATLAB 程序

```
>> A=[3  4  5;6  8  10;12  3  8];b=[1  2  3]';x1=A\b
```

运行后将屏幕显示的结果整理得

```
x1 = (Inf  Inf  Inf)'
```

因为 $\boldsymbol{A}$ 是奇异矩阵,所以 $\boldsymbol{A}$ 没有逆矩阵,故解法 2 无解.

(3) **解法 1** 输入 MATLAB 程序

```
>> A=[3  4  5;7  8  9;12  3  8];b=[4  2  3]';
[Q R] = qr(A);x = R\(R'\(A'*b)), r = b - A*x;
e = R\(R'\(A'*r)); x1 = x + e
```

运行后屏幕将显示结果

```
x =                 x1 =
    -1.8214             -1.8214
    -2.8571             -2.8571
     4.1786              4.1786
```

解法 2 输入 MATLAB 程序

```
>> x=A\b
```

运行后屏幕显示的结果与解法 1 结果相同.

5.6.3 带原点位移的 QR 方法

由 QR 算法产生的迭代序列 $\{\boldsymbol{A}_k\}$ 的收敛速度较慢,为了加快收敛速度,采取下面的带原点位移的 QR 方法.

(一) 带原点位移的 QR 方法的步骤

所谓的带原点位移的 QR 方法就是选择数列 $\{s_k\}$,按下述方法构造矩阵序列 $\{\boldsymbol{A}_k\}$,即

步骤 1 取 $\boldsymbol{A}=\boldsymbol{A}_1$.

步骤 2 选择适当的数 s_k,将矩阵 $\boldsymbol{A}_k-s_k\boldsymbol{E}$ 进行 QR 分解,即

$$\boldsymbol{A}_k-s_k\boldsymbol{E}=\boldsymbol{Q}_k\boldsymbol{R}_k, k=1,2,\cdots,n, \tag{5.59}$$

我们称 s_k 为第 k 次迭代的**原点位移量**,简称为**位移量**.

步骤 3 构造新矩阵,令

$$A_{k+1}=R_kQ_k+s_kE. \tag{5.60}$$

选择适当的位移量 s_{k+1}，将矩阵 $A_{k+1}-s_{k+1}E$ 进行 QR 分解，即

$$A_{k+1}-s_{k+1}E=Q_{k+1}R_{k+1}, k=1,2,\cdots,n.$$

步骤 4 根据(5.59)和(5.60)式，有

$$A_{k+1}=Q_k^{\mathrm{T}}A_kQ_k=\widetilde{Q}_k^{\mathrm{T}}A\widetilde{Q}_k,$$

其中 $\widetilde{Q}_k=Q_1Q_2\cdots Q_k$.

步骤 5 矩阵 $(A-s_1E)(A-s_2E)\cdots(A-s_kE)\equiv\varphi(A)$ 有 QR 分解式

$$\prod_{i=1}^{k}(A-s_iE)=\widetilde{Q}_k\widetilde{R}_k,$$

其中 $\widetilde{R}_k=R_k\cdots R_2R_1$.

步骤 6 带原点位移的 QR 方法变换一步的计算如下：

首先用正交变换的左变换将 A_k-s_kE 化为上三角形矩阵，即

$$P_{n-1}\cdots P_2P_1(A_k-s_kE)=R_k,$$

其中 $Q_k^{\mathrm{T}}=P_{n-1}\cdots P_2P_1$ 为一系列平面旋转矩阵的乘积. 于是

$$A_{k+1}=P_{n-1}\cdots P_2P_1(A_k-s_kE)P_1^{\mathrm{T}}P_2^{\mathrm{T}}\cdots P_{n-1}^{\mathrm{T}}+s_kE.$$

(二) 选择数列 $\{s_k\}$ 的方法

在每一次迭代中都需要选择位移量 s_k. 在这里介绍两种关于选择数列 $\{s_k\}$ 的方法. 为了介绍方便，将这两种方法分别称作**最末元位移法和求根位移法**.

在每一次迭代中，由带原点位移的 QR 方法的步骤可以求出一个迭代矩阵 A_k，用迭代矩阵 A_k 中右下角的四个元构成矩阵，记作

$$C_k=\begin{pmatrix} a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{pmatrix} \xlongequal{\text{记作}} \begin{pmatrix} d_{n-1} & b_{n-1} \\ b_n & d_n \end{pmatrix}. \tag{5.61}$$

方法 1 最末元位移法. 此方法就是取

$$s_k=a_{nn}^{(k)}. \tag{5.62}$$

方法 2 求根位移法. 此方法就是首先计算矩阵 C_k 的特征值 x_1,x_2，其中 x_1,x_2 是下列方程的根

$$x^2-(d_n+d_{n-1})x+d_{n-1}d_n-b_{n-1}b_n=0, \tag{5.63}$$

然后选择(5.59)式中的 s_k 为最接近 d_n 的(5.63)式的根.

可以证明，对称三角形矩阵(5.59)的位移 QR 算法收敛，且为 3 阶收敛.

5.6.4 位移的 QR 方法计算实矩阵的特征值及其 MATLAB 程序

如果 n 阶实矩阵 A 的特征值为 λ_i $(i=1,2,\cdots,n)$，则 λ_i-s_k $(i=1,2,\cdots,n)$ 是 $B=A-s_kE$ 的特征值. 将上述思想融入下面用带原点位移的 QR 方法计算实矩阵的特征值的步骤：

步骤1 用MATLAB程序 Householdrenl(A)或 house(A)将 n 阶实矩阵 $\boldsymbol{A}$ 化为上海森伯格(Hessenberg)矩阵(或对称三对角矩阵)$\boldsymbol{A}_1$.

步骤2 用MATLAB程序选择位移量 s_k.

在每一次迭代中,都需要选择位移量 s_k. 选择 s_k 的方法有几种,在我们这里介绍两种方法如下:

方法1 最末元位移法用MATLAB程序 sk = Ak(n,n),选择位移量 $s_k = a_{nn}^{(k)}$. 其中 $a_{nn}^{(k)} = d_n$ 是第 k 次迭代的迭代矩阵 $\boldsymbol{A}_k(k=1,2,\cdots,n,\cdots)$ 中的第 n 行第 n 列上的元,即是(5.61)式中的 d_n;

方法2 求根位移法用下面的MATLAB程序选择位移量 s_k.

① 先用根据(5.61)式和(5.63)式编写的程序

```
x = solve('x^2 -(d4 +d3) * x +d3 * d4 - b3 * b4 =0')
```

求出方程(5.63)的根 x_1, x_2;

② 再用下面的程序选择位移量 s_k,即

```
X1 = abs(x1 - d4); X2 = abs(x2 - d4);
if X1 >X2
    sk = x1;
else
    sk = x2;
    end
```

运行后输出的结果 s_k 就是(5.59)式中的最接近(5.61)式中 d_n 的方程(5.63)的根.

步骤3 用MATLAB程序进行左变换计算.

选择平面旋转矩阵 $\boldsymbol{P}_{12}, \boldsymbol{P}_{23}, \cdots, \boldsymbol{P}_{n-1,n}$,依次左乘矩阵$(\boldsymbol{A}_k - s_k\boldsymbol{E})$,可将$(\boldsymbol{A}_k - s_k\boldsymbol{E})$化为 n 阶上三角形矩阵,即

$$\boldsymbol{P}_{n-1,n}\cdots\boldsymbol{P}_{23}\boldsymbol{P}_{12}(\boldsymbol{A}_k - s_k\boldsymbol{E}) = \boldsymbol{R}_k,$$

其中 $\boldsymbol{R}_k$ 为 n 阶上三角形矩阵. 从而有

$$\boldsymbol{A}_k - s_k\boldsymbol{E} = \boldsymbol{P}_{12}^{\mathrm{T}}\boldsymbol{P}_{23}^{\mathrm{T}}\cdots\boldsymbol{P}_{n-1,n}^{\mathrm{T}}\boldsymbol{R}_k. \tag{5.64}$$

可以用下面的MATLAB程序进行左变换计算,即

```
[m, m] = size(Ak),
P = P12 '* P23 '* ... * Pn -1n '  % P = P'12 P'23 ··· P'n-1,n;
Pk = P ';Rk = Pk * (Ak - sk * eye(m))
```

步骤4 用MATLAB程序进行右变换计算.

根据(5.59)和(5.60),令

$$\boldsymbol{A}_{k+1} = \boldsymbol{R}_k\boldsymbol{P}_{12}^{\mathrm{T}}\boldsymbol{P}_{23}^{\mathrm{T}}\cdots\boldsymbol{P}_{n-1,n}^{\mathrm{T}} + s_k\boldsymbol{E},$$

然后选择适当的数 s_{k+1} 替换(5.62)式中的 s_k,得

$$\boldsymbol{A}_{k+1} - s_{k+1}\boldsymbol{E} = \boldsymbol{R}_k\boldsymbol{P}_{12}^{\mathrm{T}}\boldsymbol{P}_{23}^{\mathrm{T}}\cdots\boldsymbol{P}_{n-1,n}^{\mathrm{T}}.$$

因为矩阵 $\boldsymbol{R}_k\boldsymbol{P}_{12}^{\mathrm{T}}\boldsymbol{P}_{23}^{\mathrm{T}}\cdots\boldsymbol{P}_{n-1,n}^{\mathrm{T}}$是上海森伯格矩阵,所以经过计算得到的

$$\boldsymbol{A}_{k+1}=\boldsymbol{R}_k\boldsymbol{P}_{12}^{\mathrm{T}}\boldsymbol{P}_{23}^{\mathrm{T}}\cdots\boldsymbol{P}_{n-1,n}^{\mathrm{T}}+s_{k+1}\boldsymbol{E} \tag{5.65}$$

也是上海森伯格矩阵.

可以用下面的 MATLAB 程序进行左变换计算,即

```
Ak = Rk * P + sk * eye(m)
```

显然,每一次左变换仅改变矩阵的两行,而每一次右变换仅改变矩阵的两列. 为了节省存储量. 左变换和右变换可同时进行.

步骤 5 反复应用步骤 2 至步骤 4,求矩阵 $\boldsymbol{A}$ 的近似特征值.

反复应用步骤 2 至步骤 4,就产生一个正交相似于上海森伯格矩阵 $\boldsymbol{A}$ 的序列$\{\boldsymbol{A}_k\}$,选择位移量 s_k 的方法不同,则矩阵 $\boldsymbol{A}$ 的近似特征值的计算方法就不同. 在我们这里介绍两种方法如下:

方法 1 如果用最末元位移法选择位移量 $s_k=a_{nn}^{(k)}$,反复应用步骤 2 至步骤 4,当$|a_{n,n-1}^{(k)}|$充分小时,则数 $\lambda_n=a_{nn}^{(k)}$为矩阵 $\boldsymbol{A}$ 的近似特征值,且

$$\boldsymbol{A}_k=\left(\begin{array}{cccccc:c} * & * & \cdots & * & * & * \\ * & * & \cdots & * & * & * \\ & * & \ddots & \vdots & \vdots & \vdots \\ & & \ddots & * & * & * \\ & & & * & * & * \\ \hdashline & & & & a_{n,n-1}^{(k)} & \lambda_n \end{array}\right)_{n\times n}\approx\begin{pmatrix} \boldsymbol{B}_{(n-1)\times(n-1)} & \boldsymbol{C}_{(n-1)\times 1} \\ \boldsymbol{O}_{1\times(n-1)} & \lambda_n \end{pmatrix},$$

其中判别$|a_{n,n-1}^{(k)}|$充分小的准则是:

① $|a_{n,n-1}^{(k)}|\le\varepsilon\|\boldsymbol{A}\|_\infty$;

② 取 $\varepsilon=10^{-t}$,$t+1$ 是计算过程中有效数字的个数,将 $a_{n,n-1}^{(k)}$与相邻元进行比较,即

$$|a_{n,n-1}^{(k)}|\le\varepsilon\min\left(|a_{n,n}^{(k)}|,|a_{n-1,n-1}^{(k)}|\right).$$

方法 2 如果用求根位移法选择位移量 s_k,反复应用步骤 2 至步骤 4,当$a_{n,n-1}^{(k)}\approx 0$ 时,则数 $\lambda_j=s_1+s_2+\cdots+s_{k_j}$为矩阵 $\boldsymbol{A}$ 的近似特征值.

步骤 6 采用逐步收缩的方法. 计算的矩阵 $\boldsymbol{A}$ 的一个特征值后,可将矩阵降阶,取 $n-1$ 阶主子矩阵 $\boldsymbol{B}_{(n-1)\times(n-1)}$,继续对 $\boldsymbol{B}_{(n-1)\times(n-1)}$应用上述方法,就可以逐步求出 $\boldsymbol{A}$ 的其余近似特征值.

值得注意的是:上述的带原点位移的 QR 方法可以计算实矩阵 $\boldsymbol{A}$ 的所有实特征值,但是不能计算 $\boldsymbol{A}$ 的复特征值. 如果矩阵 $\boldsymbol{A}$ 有复特征值,则上述带原点位移的 QR 方法不收敛. 在这种情形下,可应用对上述过程作了修改的双重步 QR 方法(参见曹志浩等人编著的《矩阵计算和方程组求根》).

5.6.5 最末元位移 QR 方法计算实对称矩阵特征值及其 MATLAB程序

如果 n 阶实矩阵 $\boldsymbol{A}$ 的特征值为 $\lambda_i\ (i=1,2,\cdots,n)$，则 $\lambda_i-s_k\ (i=1,2,\cdots,n)$ 是 $\boldsymbol{B}=\boldsymbol{A}-s_k\boldsymbol{E}$ 的特征值. 将上述思想融入下面用带原点位移的 QR 方法计算实矩阵的特征值. 带原点位移的 QR 方法特别适用于计算实对称矩阵的特征值. 在这里为了介绍方便，我把用最末元位移法选择位移量 s_k 的带原点位移的 QR 方法求 $\boldsymbol{A}$ 的所有近似特征值的方法称作**最末元位移 QR 方法**.

（一）最末元位移 QR 方法的一般步骤

如果 n 阶实矩阵 $\boldsymbol{A}$ 是实对称矩阵，则可以用最末元位移 QR 方法计算 $\boldsymbol{A}$ 的所有近似特征值，其步骤如下：

步骤 1 用 MATLAB 程序选择位移量 s_k.

在每一次迭代中，都需要选择位移量 s_k. 选择 s_k 的方法有几种，在这里采用最末元位移法，可用 MATLAB 程序 `sk = Ak(n,n)` 选择位移量 $s_k=a_{nn}^{(k)}$. 其中 $a_{nn}^{(k)}=d_n$ 是第 k 次迭代的迭代矩阵 $\boldsymbol{A}_k^{*}\ (k=1,2,\cdots,n,\cdots)$ 中的第 n 行第 n 列上的元，即是(5.61)式中的 d_n；

步骤 2 用 MATLAB 程序计算 $\boldsymbol{A}_k-s_k\boldsymbol{E}$，然后将 $\boldsymbol{A}_k-s_k\boldsymbol{E}$ 进行 QR 分解，即

$$\boldsymbol{A}_k-s_k\boldsymbol{E}=\boldsymbol{Q}_k\boldsymbol{R}_k,\ k=1,2,\cdots,n,\cdots.$$

可以用下面的 MATLAB 程序完成，即

```
[m m] = size(Ak); B1 = Ak - sk * eye (m); [Qk,Rk] = qr(B1);
```

步骤 3 构造新矩阵

$$\boldsymbol{A}_{k+1}=\boldsymbol{R}_k\boldsymbol{Q}_k+s_k\boldsymbol{E},$$

可以用下面的 MATLAB 程序完成上式的计算，即

```
Ak1 = Rk * Qk + sk * eye(m).
```

步骤 4 求矩阵 $\boldsymbol{A}$ 的近似特征值. 反复应用步骤 1 至步骤 3，取 $\varepsilon=10^{-t}$，$t+1$ 是计算过程中有效数字的个数，当 $|a_{n,n-1}^{(k)}|\leqslant\varepsilon\min\ (|a_{n,n}^{(k)}|,|a_{n-1,n-1}^{(k)}|)$ 或 $|a_{n,n-1}^{(k)}|\leqslant\varepsilon\|\boldsymbol{A}\|_\infty$ 时，则数 $\lambda_n=a_{nn}^{(k)}$ 为矩阵 $\boldsymbol{A}$ 的近似特征值，且记

$$\boldsymbol{A}_k=\begin{pmatrix}\boldsymbol{B}_{(n-1)\times(n-1)} & \boldsymbol{C}_{(n-1)\times 1}\\ \boldsymbol{O}_{1\times(n-1)} & \lambda_n\end{pmatrix}.$$

步骤 5 求出 $\boldsymbol{A}$ 其余的近似特征值. 采用逐步收缩的方法，继续对 $\boldsymbol{B}_{(n-1)\times(n-1)}$ 应用最末元位移 QR 方法，就可以逐步求出 $\boldsymbol{A}$ 其余的近似特征值.

例 5.6.4 用最末元位移 QR 方法求实对称矩阵 $\boldsymbol{A}$ 的全部特征值，精度为 10^{-5}. 并将计算结果与 $\boldsymbol{A}$ 全部真特征值比较. 其中

$$A=\begin{pmatrix}5 & 2 & 2 & 1\\2 & -4 & 1 & 1\\2 & 1 & 3 & 1\\1 & 1 & 1 & 2\end{pmatrix}.$$

解 (1) 首先求 A 第一个特征值的近似值 λ_1.

① 因为 $|a_{n,n-1}^{(1)}|>10^{-5}\min(|a_{n,n}^{(1)}|,|a_{n-1,n-1}^{(1)}|)$, $n=4$,所以**选择位移量** $s_1=a_{4\times4}^{(1)}=2$ 为 A 的第 4 行第 4 列的元.

② 用 MATLAB 程序计算 $B_1=A_1-s_1E$,然后将 $B_1=A_1-s_1E$ 进行 QR 分解,即

$$A_1-s_1E=Q_1R_1,$$

由 MATLAB 程序

```
>> A =[5,2,2,1;2,-4,1,1;2,1,3,1;1,1,1,2]; [m,m] = size(A);
s1 = A(4,4), B1 = A - s1 * eye (m), [Q1,R1] = qr(B1),
```

运行后输出结果

```
s1 =
     2
B1 =
     3          2          2          1
     2         -6          1          1
     2          1          1          1
     1          1          1          0
Q1 =
-0.70710678118655   0.38807526285317   0.12674485010490  -0.57735026918963
-0.47140452079103  -0.87963726246718   0.06337242505245                  0
-0.47140452079103   0.20697347352169  -0.63372425052448   0.57735026918963
-0.23570226039552   0.18110178933148   0.76046910062937   0.57735026918963
R1 =
-4.24264068711929  0.70710678118655  -2.59272486435067  -1.64991582276861
                0  6.44204936336256   0.28458852609232  -0.28458852609232
                0                 0   0.44360697536713  -0.44360697536713
                0                 0                  0   0.00000000000000
```

③ 构造新矩阵 $A_2=R_1Q_1+s_1E$. 用 MATLAB 程序

```
>> A2 = R1 * Q1 + s1 * eye(m),
r2 = abs(A2(m,m-1)) - 10^(-5) * min(abs(A2(m,m)), abs(A2(m-1,m
-1)))
```

完成上式的计算,得

```
A2 =
 6.27777777777778  -3.10388935193069  -0.10455916682125  0.00000000000000
-3.10388935193069  -3.65930388219545   0.01147685957127  0.00000000000000
-0.10455916682125   0.01147685957127   1.38152610441767  0.00000000000000
 0.00000000000000   0.00000000000000   0.00000000000000  2.00000000000000
r2 =
     -1.381526104379678e-005
```

④ 因为 $|a_{n,n-1}^{(2)}|<10^{-5}\min(|a_{n,n}^{(2)}|,|a_{n-1,n-1}^{(2)}|)$,$n=4$,所以精度为 10^{-5} 的 $\boldsymbol{A}$ 的第一个特征值的近似值 $\lambda_1\approx2.000\ 00$.

(2) 再求 $\boldsymbol{A}$ 第二个特征值的近似值 λ_2.

① 取矩阵 $\boldsymbol{A}_2$ 的 3 阶主子矩阵 $\boldsymbol{A}_3$,

用 MATLAB 程序

```
>> A3 = A2(1:3,1:3), [m1,m1] = size(A3);
r3 = abs(A3(m1,m1 -1)) -10^( -5) * min(abs(A3(m1,m1)), abs(A3(m1
-1,m1 -1)))
```

完成上式的计算,得

```
A3 =
     6.27777777777778  -3.10388935193069  -0.10455916682125
    -3.10388935193069  -3.65930388219545   0.01147685957127
    -0.10455916682125   0.01147685957127   1.38152610441767
r3 =
   0.01151345261010
```

② 重复上述步骤,即因为 $|a_{n,n-1}^{(3)}|>10^{-5}\min(|a_{n,n}^{(3)}|,|a_{n-1,n-1}^{(3)}|)$,$n=3$,所以继续迭代. 选择位移量 $s_2=a_{3\times3}^{(3)}=1.381\ 526\ 104\ 417\ 67$ 为 $\boldsymbol{A}_3$ 的第 3 行第 3 列的元.

③ 计算 $\boldsymbol{B}_2=\boldsymbol{A}_3-s_2\boldsymbol{E}$,并将其进行 QR 分解,即 $\boldsymbol{A}_3-s_2\boldsymbol{E}=\boldsymbol{Q}_2\boldsymbol{R}_2$,由 MATLAB 程序

```
>> s2 = A3(m1,m1), B2 = A3 - s2 * eye (m1), [Q2,R2] = qr(B2),
```

运行后输出结果

```
s2 =
    1.38152610441767
B2 =
    4.89625167336011  -3.10388935193069  -0.10455916682125
   -3.10388935193069  -5.04082998661312   0.01147685957127
   -0.10455916682125   0.01147685957127                  0
```

```
Q2 =
  -0.84445320114929  -0.53537837009187   0.01639487439677
   0.53532568873289  -0.84460953959679  -0.00781873421730
   0.01803324849744   0.00217404228940   0.99983502413586
R2 =
  -5.79813264571247  -0.07718952005739   0.09443918088619
                  0   5.91931326753920   0.04628525123242
                  0                  0  -0.00180396892170
```

④ 构造新矩阵 $\boldsymbol{A}_4=\boldsymbol{R}_2\boldsymbol{Q}_2+s_2\boldsymbol{E}$. 用 MATLAB 程序

```
>>A4 = R2 * Q2 + s2 * eye(m1),
r4 = abs(A4(m1,m1  1)) - 10^(  5) * min(abs(A4(m1,m1)), abs(A4(m1
-1,m1 -1))),
```

完成上式的计算,得

```
A4 =
   6.23815929000691   3.16959512520840  -0.00003253141985
   3.16959512520840  -3.61788172311421  -0.00000392190472
  -0.00003253141985  -0.00000392190472   1.37972243310730
r4 =
  -9.875319606520319e-006
```

⑤ 因为 $|a_{n,n-1}^{(4)}|<10^{-5}\min(|a_{n,n}^{(4)}|,|a_{n-1,n-1}^{(4)}|)$, $n=3$,所以精度为 10^{-5} 的 $\boldsymbol{A}$ 的第二个特征值的近似值 $\lambda_2\approx1.379\ 72$.

(3) 再求 $\boldsymbol{A}$ 第三个特征值的近似值 λ_3.

① 取矩阵 $\boldsymbol{A}_4$ 的 2 阶主子矩阵 $\boldsymbol{A}_5$,用 MATLAB 程序

```
>>A5 = A4(1:2,1:2), [m2,m2] = size(A5);
r5 = abs(A5(m2,m2 -1)) -10^(-5) * min(abs(A5(m2,m2)), abs(A5(m2
-1,m2 -1)))
```

完成上式的计算,得

```
A5 =
   6.23815929000691   3.16959512520840
   3.16959512520840  -3.61788172311421
r5 =
   3.16955894639117
```

② 因为 $|a_{n,n-1}^{(5)}|>10^{-5}\min(|a_{n,n}^{(5)}|,|a_{n-1,n-1}^{(5)}|)$, $n=2$,所以继续迭代. 选择位移量 $s_3=a_{2\times2}^{(5)}=-3.617\ 881\ 723\ 114\ 21$ 为 $\boldsymbol{A}_5$ 的第 2 行第 2 列的元. 重复上述步骤,即输入 MATLAB 程序

```
>>s3 = A5(m2,m2), B3 = A5 - s3 * eye (m2), [Q3,R3] = qr(B3),
A6 = R3 * Q3 + s3 * eye(m2),
```

```
R6 = A6(m2,m2 -1) -10^( -5) * min(abs(A6(m2,m2)),
abs(A6(m2 -1,m2 -1))),s4 = A6(m2,m2), B4 = A6 - s4 * eye (m2), [Q4,
R4] = qr(B4),
A7 = R4 * Q4 + s4 * eye(m2),
R7 = A7(m2,m2 -1) -10^( -5) * min(abs(A7(m2,m2)), abs(A7(m2 -1,
m2 -1))),
```

运行后输出结果

```
s3 =
  -4.54165290634115
B3 =
   11.70358337957500    0.29707472151000
    0.29707472151000                   0
Q3 =
  -0.99967800146784   -0.02537505431063
  -0.02537505431063    0.99967800146784
R3 =
 -11.70735313009838   -0.29697906388573
                  0   -0.00753828719263
A6 =
    7.16946633310907    0.00019128444692
    0.00019128444692   -4.54918876621637
R6 =
    1.457925592600288e-004
s4 =
  -4.54918876621637
B4 =
   11.71865509932544    0.00019128444692
    0.00019128444692                   0
Q4 =
  -0.99999999986678   -0.00001632307166
  -0.00001632307166    0.99999999986678
R4 =
 -11.71865510088661   -0.00019128444690
                  0   -0.00000000312235
A7 =
    7.16946633623142    0.00000000000005
    0.00000000000005   -4.54918876933872
R7 =
```

-4.549188764242086e-005

因为$|a_{n,n-1}^{(7)}|<10^{-5}\min(|a_{n,n}^{(7)}|,|a_{n-1,n-1}^{(7)}|)$，$n=2$，所以精度为$10^{-5}$的$\boldsymbol{A}$的第三个特征值的近似值$\lambda_3\approx-4.549\ 19$.

（4）最后求$\boldsymbol{A}$第四个特征值的近似值λ_4.

因为矩阵$\boldsymbol{A}_7$的1阶主子矩阵$\boldsymbol{A}_8=7.169\ 466\ 336\ 231\ 42$，所以精度为$10^{-5}$的$\boldsymbol{A}$的第四个特征值的近似值$\lambda_4\approx7.169\ 47$.

综上所述，计算机计算的精度为10^{-5}的$\boldsymbol{A}$的第四个特征值的近似值分别为

$\lambda_1\approx2.000\ 000\ 000\ 000\ 00$，$\lambda_2\approx1.379\ 722\ 433\ 107\ 30$，

$\lambda_3\approx-4.549\ 188\ 769\ 338\ 72$，$\lambda_4\approx7.169\ 466\ 336\ 231\ 42$.

即 $\lambda_1\approx2.000\ 00$，$\lambda_2\approx1.379\ 72$. $\lambda_3\approx-4.549\ 19$，$\lambda_4\approx7.169\ 47$

实际上，根据程序 b=eig(A) 计算出4阶实对称矩阵$\boldsymbol{A}$的全部真特征值为

$\lambda_1=2.000\ 000\ 000\ 000\ 00$，$\lambda_2=1.379\ 722\ 432\ 931\ 84$，

$\lambda_3=-4.549\ 188\ 769\ 343\ 66$，$\lambda_4=7.169\ 466\ 336\ 411\ 81$.

由此可见，特征值的近似值的精度都达到10^{-8}.

（二）用最末元位移 QR 方法求实对称矩阵 $\boldsymbol{A}$ 的全部特征值的 MATLAB 主程序

如果n阶矩阵$\boldsymbol{A}$是实对称矩阵，则用带原点位移的QR方法中的最末元位移QR方法计算$\boldsymbol{A}$的所有近似特征值可以用下面的MATLAB程序完成，并且给出全部计算过程的信息.

> **用最末元位移 QR 方法求实对称矩阵 $\boldsymbol{A}$ 的全部特征值的 MATLAB 主程序**
>
> 输入的量：$\boldsymbol{A}$是n阶实对称矩阵，t是精度$\varepsilon=10^{-t}$的指数，$MAX1$是最大迭代次数.
>
> 输出的量：i表示求第i个特征值，k是迭代次数，s_k是原点位移量，$\boldsymbol{Q}_k$和$\boldsymbol{R}_k$是$\boldsymbol{B}_k$的QR分解，$\boldsymbol{A}_t$迭代矩阵，$tzgk$是矩阵$\boldsymbol{A}$的特征值的近似值，矩阵$\boldsymbol{B}$是m阶矩阵$\boldsymbol{A}_t$的$(m-1)$阶主子矩阵及其相关信息. 当迭代矩阵$\boldsymbol{A}_k$降一阶时，给出相关的提示.

根据最末元位移QR方法求实对称矩阵$\boldsymbol{A}$全部特征值的步骤，现提供用最末元位移QR方法求实对称矩阵$\boldsymbol{A}$的全部特征值的MATLAB主程序如下，取名为qr4.m.

```
function tzg=qr4(A,t,max1)
  [n,n]=size(A); k=0;Ak=A;tzg=zeros(n); state=1;
  for i=1:n;
  while((k<=max1)&(state==1)&(n>1))
    b1=abs(Ak(n,n-1)); b2=abs(Ak(n,n)); b3=abs(Ak(n-1,n-1));
    b4=min(b2, b3); jd=10^(-t); jd1=jd*b4;
```

```
        if(b1 > = jd1)
            sk = Ak(n,n); Bk = Ak - sk * eye(n); [Qk,Rk] = qr(Bk); .
            At = Rk * Qk + sk * eye(n); k = k + 1;tzgk = Ak(n,n);
            disp('请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点
位移量,')
            disp('    Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk
* Qk + sk * eye(n)迭代矩阵:')
            i,state = 1;k,sk,Bk,Qk,Rk,At,Ak = At;
        else
          disp('请注意:i 表示求第 i 个特征值,tzgk 是矩阵 A 的特征值的近似值,
k 是迭代次数,')
          disp('    下面的矩阵 B 是 m 阶矩阵 At 的(m - 1)阶主子矩阵,即 At 降一
阶.')
          i,tzgk = Ak(n,n),tzg(n,1) = tzgk;
          k = k,sk,Ak;B = Ak(1:n - 1,1:n - 1),
          Ak = B;n = n - 1;state = = 1; i = i + 1;
        end
      end
    end
    tzg(1,1) = Ak;tzg = sort(tzg(:,1));tzgk = Ak
            disp('请注意:n 阶实对称矩阵 A 的全部真特征值 lamoda 和至少含 t
个有效数字的近似特征值 tzg 如下:')
    lamoda = sort(eig(A))
```

例 5.6.5 用最末元位移 QR 方法求下列实对称矩阵的全部近似特征值,并将计算结果与 $\boldsymbol{A}$ 全部真特征值比较,其中

(1) $\boldsymbol{A}=\begin{pmatrix}5&2&2&1\\2&-4&1&1\\2&1&3&1\\1&1&1&2\end{pmatrix}$,精度为 $\varepsilon=10^{-5}$;

(2) $\boldsymbol{A}=\begin{pmatrix}12&-56&3&-14&-90&-41\\-56&71&23&61&-9&-21\\3&23&53&12&-72&51\\-14&61&12&73&23&21\\-90&-9&-72&23&-34&-61\\-41&-21&51&21&-61&-52\end{pmatrix}$,精度为 $\varepsilon=10^{-4}$.

解 (1) 首先保存用最末元位移 QR 方法求实对称矩阵 $\boldsymbol{A}$ 的全部特征值的 MATLAB 主程序为 M 文件,取名为 qr4.m. 在 MATLAB 工作窗口输入程序

```
>> A=[5  2  2  1;2  -4  1  1;2  1  3  1;1  1  1  2]; tzg=qr4(A,5,100)
```

运行后屏幕显示结果

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,

Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)

迭代矩阵:

```
i =
    1
k =
    1
sk =
    2
Bk =
    3     2     2     1
    2    -6     1     1
    2     1     1     1
    1     1     1     0
Qk =
 -0.70710678118655    0.38807526285317    0.12674485010490   -0.57735026918963
 -0.47140452079103   -0.87963726246718    0.06337242505245                   0
 -0.47140452079103    0.20697347352169   -0.63372425052448    0.57735026918963
 -0.23570226039552    0.18110178933148    0.76046910062937    0.57735026918963
Rk =
 -4.24264068711929    0.70710678118655   -2.59272486435067   -1.64991582276861
                 0    6.44204936336256    0.28458852609232   -0.28458852609232
                 0                   0    0.44360697536713   -0.44360697536713
                 0                   0                   0    0.00000000000000
At =
  6.27777777777778   -3.10388935193069   -0.10455916682125    0.00000000000000
 -3.10388935193069   -3.65930388219545    0.01147685957127    0.00000000000000
 -0.10455916682125    0.01147685957127    1.38152610441767    0.00000000000000
 -0.00000000000000    0.00000000000000    0.00000000000000    2.00000000000000
```

请注意:i 表示求第 i 个特征值,tzgk 是矩阵 A 的特征值的近似值,k 是迭代次数,

下面的矩阵 B 是 m 阶矩阵 At 的(m-1)阶主子矩阵,即 At 降一阶.

```
i =
    1
tzgk =
    2.00000000000000
k =
```

```
        1
sk =
        2
B =
 6.27777777777778  -3.10388935193069  -0.10455916682125
-3.10388935193069  -3.65930388219545   0.01147685957127
-0.10455916682125   0.01147685957127   1.38152610441767
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,

Bk = Ak sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)

迭代矩阵:

```
i =
        2
k =
        2
sk =
    1.38152610441767
Bk =
 4.89625167336011  -3.10388935193069  -0.10455916682125
-3.10388935193069  -5.04082998661312   0.01147685957127
-0.10455916682125   0.01147685957127                  0
Qk =
-0.84445320114929  -0.53537837009187   0.01639487439677
 0.53532568873289  -0.84460953959679  -0.00781873421730
 0.01803324849744   0.00217404228940   0.99983502413586
Rk =
-5.79813264571247  -0.07718952005739   0.09443918088619
                0   5.91931326753920   0.04628525123242
                0                  0  -0.00180396892170
At =
 6.23815929000691   3.16959512520840  -0.00003253141985
 3.16959512520840  -3.61788172311421  -0.00000392190472
-0.00003253141985  -0.00000392190472   1.37972243310730
```

请注意:i 表示求第 i 个特征值,tzgk 是矩阵 A 的特征值的近似值,k 是迭代次数,

下面的矩阵 B 是 m 阶矩阵 At 的(m-1)阶主子矩阵,即 At 降一阶.

```
i =
        2
tzgk =
    1.37972243310730
```

```
k =
      2
sk =
    1.38152610441767
B =
    6.23815929000691    3.16959512520840
    3.16959512520840   -3.61788172311421
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,

Bk - Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)

迭代矩阵:

```
i -
      3
k =
      3
sk =
   -3.61788172311421
Bk =
    9.85604101312112    3.16959512520840
    3.16959512520840                   0
Qk =
   -0.95198403663348   -0.30614766697629
   -0.30614766697629    0.95198403663348
Rk =
  -10.35315786173815   -3.01740396178969
                   0   -0.97036415284199
At =
    7.16193047323385    0.29707472151000
    0.29707472151000   -4.54165290634115
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,

Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)

迭代矩阵:

```
i =
      3
k =
      4
sk =
   -4.54165290634115
Bk =
```

```
   11.70358337957500    0.29707472151000
    0.29707472151000                   0
Qk =
   -0.99967800146784   -0.02537505431063
   -0.02537505431063    0.99967800146784
Rk =
  -11.70735313009838   -0.29697906388573
                   0   -0.00753828719263
At =
    7.16946633310907    0.00019128444692
    0.00019128444692   -4.54918876621637
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,

Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)

迭代矩阵:

```
i =
     3
k =
     5
sk =
  -4.54918876621637
Bk =
   11.71865509932544    0.00019128444692
    0.00019128444692                   0
Qk =
   -0.99999999986678   -0.00001632307166
   -0.00001632307166    0.99999999986678
Rk =
  -11.71865510088661   -0.00019128444690
                   0   -0.00000000312235
At =
    7.16946633623142    0.00000000000005
    0.00000000000005   -4.54918876933872
```

请注意:i 表示求第 i 个特征值,tzgk 是矩阵 A 的特征值的近似值,k 是迭代次数,

下面的矩阵 B 是 m 阶矩阵 At 的(m-1)阶主子矩阵,即 At 降一阶.

```
i =
     3
tzgk =
  -4.54918876933872
```

```
k =
      5
sk =
   -4.54918876621637
B =
    7.16946633623142
tzgk =
    7.16946633623142
```

请注意:n 阶实对称矩阵 A 的全部真特征值 lamoda 和至少含 t 个有效数字的近似特征值 tzg 如下:

```
lamoda =
   -4.54918876934366
    1.37972243293184
    2.00000000000000
    7.16946633641181
tzg =
   -4.54918876933872
    1.37972243310730
    2.00000000000000
    7.16946633623142
```

(2) 在 MATLAB 工作窗口输入程序

```
>> A=[12  56 3 -14 -90  41; -56 71 23 61 -9 -21;3 23 53 12 -72 51; -14 61 12 73 23 21; -90 -9 -72 23 -34 -61; -41 -21 51 21 -61 -52],
tzg=qr4(A,4,100)
```

运行后屏幕显示结果

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Bk=Ak-sk*eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At=Rk*Qk+sk*eye(n)迭代矩阵:

```
i =
      1
k =
      1
sk =
    -52
Bk =
      64    -56      3    -14    -90    -41
     -56    123     23     61     -9    -21
       3     23    105     12    -72     51
     -14     61     12    125     23     21
```

```
   -90     -9    -72    23    18    -61
   -41    -21     51    21   -61      0
Qk =
 -0.4877    0.1531    0.0110   -0.3210   -0.7966   -0.0302
  0.4268   -0.6902    0.0293    0.2307   -0.4943    0.2076
 -0.0229   -0.1854   -0.7498    0.0708   -0.0367   -0.6298
  0.1067   -0.3999    0.0569   -0.8801    0.2157   -0.0653
  0.6859    0.4443    0.2359   -0.1349   -0.2604   -0.4365
  0.3125    0.3294   -0.6148   -0.2143   -0.0731    0.6038
Rk =
-131.2174   73.0543  -26.2160   68.2608   37.4417  -29.7293
        0 -133.0416  -54.8715  -79.3230  -15.5119  -36.7393
        0         0 -125.6735   -7.7419   95.7846  -52.4985
        0         0         0  -98.2059   12.1142    1.6737
        0         0         0         0   83.5168   61.5863
        0         0         0         0         0   -9.9811
At =
  67.4514  -86.1061   51.3408   -1.6460   76.5261   -3.1187
 -86.1061   62.7258   51.6606   45.2076   57.3920   -3.2880
  51.3408   51.6606   96.6541   -3.7567  -18.1668    6.1366
  -1.6460   45.2076   -3.7567   32.4419  -24.4622    2.1386
  76.5261   57.3920  -18.1668  -24.4622  -78.2465    0.7293
  -3.1187   -3.2880    6.1366    2.1386    0.7293  -58.0267
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量, Bk = Ak - sk * eye(n),Qk **和** Rk **是** Bk **的** QR **分解**,At = Rk * Qk + sk * eye(n)**迭代矩阵:**

```
i =
      1
k =
      2
sk =
-58.0267
Bk =
 125.4782  -86.1061   51.3408   -1.6460   76.5261   -3.1187
 -86.1061  120.7526   51.6606   45.2076   57.3920   -3.2880
  51.3408   51.6606  154.6809   -3.7567  -18.1668    6.1366
  -1.6460   45.2076   -3.7567   90.4686  -24.4622    2.1386
  76.5261   57.3920  -18.1668  -24.4622  -20.2198    0.7293
  -3.1187   -3.2880    6.1366    2.1386    0.7293         0
```

```
Qk =
  -0.7052    0.1935   -0.2299   -0.3147    0.5598   -0.0039
   0.4839   -0.5340   -0.2202   -0.0946    0.6505    0.0054
  -0.2885   -0.4861   -0.7045    0.1495   -0.4009   -0.0315
   0.0093   -0.2893    0.1849   -0.8863   -0.3106   -0.0076
  -0.4301   -0.5970    0.6032    0.2892    0.0754    0.0742
   0.0175    0.0305   -0.0655   -0.0243   -0.0220    0.9967
Rk =
-177.9427   79.9237  -47.9486   35.5150  -12.4675   -1.4563
        0 -153.7038  -80.7260  -34.1371   12.1657   -2.8852
        0         0 -144.2113   -5.0943  -34.1981   -2.0472
        0         0         0  -91.6269  -16.4126    0.5252
        0         0         0         0   93.5178   -6.9543
        0         0         0         0         0   -0.1614
At =
 125.6255  -56.6838   56.2327    6.2201  -40.3402   -0.0028
 -56.6838   65.8196   91.9343   36.3234  -56.0440   -0.0049
  56.2327   91.9343   22.1411  -26.8778   56.8615    0.0106
   6.2201   36.3234  -26.8778   18.4203   27.2120    0.0039
 -40.3402  -56.0440   56.8615   27.2120  -50.8189    0.0036
  -0.0028   -0.0049    0.0106    0.0039    0.0036  -58.1877
```

请注意:i 表示求第 i 个特征值,tzgk 是矩阵 A 的特征值的近似值,k 是迭代次数,下面的矩阵 B 是 m 阶矩阵 At 的(m-1)阶主子矩阵,即 At 降一阶.

```
i =
     1
tzgk =
-58.1877
k =
     2
sk =
-58.0267
B =
 125.6255  -56.6838   56.2327    6.2201  -40.3402
 -56.6838   65.8196   91.9343   36.3234  -56.0440
  56.2327   91.9343   22.1411  -26.8778   56.8615
   6.2201   36.3234  -26.8778   18.4203   27.2120
 -40.3402  -56.0440   56.8615   27.2120  -50.8189
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Bk = Ak - sk *

eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)迭代矩阵:

```
i =
      2
k =
      3
sk =
-50.8189
Bk =
176.4443   -56.6838    56.2327     6.2201   -40.3402
-56.6838   116.6385    91.9343    36.3234   -56.0440
 56.2327    91.9343    72.9600   -26.8778    56.8615
  6.2201    36.3234   -26.8778    69.2391    27.2120
-40.3402   -56.0440    56.8615    27.2120          0
Qk =
 -0.8915     0.0982     0.2806    -0.2194    -0.2623
  0.2864    -0.6231     0.4721    -0.2454    -0.4965
 -0.2841    -0.6298     0.1645     0.3591     0.6055
 -0.0314    -0.2269    -0.3554    -0.8387     0.3432
  0.2038     0.3923     0.7383    -0.2435     0.4475
Rk =
-197.9237    45.2527   -32.0958    15.8644     2.9015
        0  -166.3791   -69.3146   -10.1284   -11.0254
        0          0   122.7118     9.9548   -38.0948
        0          0          0   -84.6267    20.1978
        0          0          0          0    82.1756
At -
147.7971   -29.8854   -42.9412     6.7762    16.7488
-29.8854    94.4890   -94.4902    27.1220    32.2359
-42.9412   -94.4902   -62.2964    44.9875    60.6674
  6.7762    27.1220    44.9875    15.2418   -20.0057
 16.7488    32.2359    60.6674   -20.0057   -14.0439
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量, Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)迭代矩阵:

```
i =
      2
k =
      4
sk =
```

```
-14.0439
Bk =
161.8411   -29.8854   -42.9412     6.7762    16.7488
-29.8854   108.5329   -94.4902    27.1220    32.2359
-42.9412   -94.4902   -48.2524    44.9875    60.6674
  6.7762    27.1220    44.9875    29.2857   -20.0057
 16.7488    32.2359    60.6674   -20.0057          0
Qk =
 -0.9462     0.0767    -0.2892    -0.1036     0.0670
  0.1747    -0.6934    -0.6522    -0.1674     0.1881
  0.2511     0.6549    -0.4023    -0.4167     0.4154
 -0.0396    -0.1839     0.3418    -0.8875    -0.2454
 -0.0979    -0.2250     0.4609    -0.0073     0.8528
Rk =
-171.0440    19.2874     4.2844    10.4203     5.8081
        0  -151.6697     8.6993    10.2944    22.3445
        0          0   136.7883   -36.9542   -57.1088
        0          0          0   -49.8308   -14.6542
        0          0          0          0    37.2997
At =
151.2612   -26.9120    41.3973     3.4091    -3.6524
-26.9120    89.8963   109.2297    12.4590    -8.3921
 41.3973   109.2297  -108.0218   -23.7870    17.1910
  3.4091    12.4590   -23.7870    30.2856    -0.2714
 -3.6524    -8.3921    17.1910    -0.2714    17.7663
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)迭代矩阵:

```
i =
      2
k =
      5
sk =
17.7663
Bk =
133.4948   -26.9120    41.3973     3.4091    -3.6524
-26.9120    72.1300   109.2297    12.4590    -8.3921
 41.3973   109.2297  -125.7881   -23.7870    17.1910
  3.4091    12.4590   -23.7870    12.5193    -0.2714
```

```
  -3.6524   -8.3921   17.1910   -0.2714         0
 Qk =
  -0.9373    0.1544    0.3111   -0.0283   -0.0060
   0.1890   -0.5278    0.8259   -0.0529   -0.0288
  -0.2907   -0.8275   -0.4500    0.1520    0.0714
  -0.0239   -0.0939   -0.1102   -0.9713    0.1873
   0.0256    0.0637    0.0801    0.1729    0.9793
 Rk =
-142.4213    6.5920   19.4103    5.7664   -3.1527
        0 -134.3143   56.1648   12.4416  -10.3350
        0         0  163.6998   20.6535  -15.7739
        0         0         0  -16.5790    3.4248
        0         0         0         0    1.4409
 At =
 146.6460  -42.2683  -48.4812    0.4847    0.0370
 -42.2683   40.3466 -138.4080    1.7756    0.0918
 -48.4812 -138.4080  -59.4436    2.1021    0.1154
   0.4847    1.7756    2.1021   34.4614    0.2491
   0.0370    0.0918    0.1154    0.2491   19.1773
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)迭代矩阵:

```
 i =
      2
 k =
      6
 sk =
 19.1773
 Bk =
 127.4686  -42.2683  -48.4812    0.4847    0.0370
 -42.2683   21.1692 -138.4080    1.7756    0.0918
 -48.4812 -138.4080  -78.6209    2.1021    0.1154
   0.4847    1.7756    2.1021   15.2841    0.2491
   0.0370    0.0918    0.1154    0.2491         0
 Qk =
  -0.8928    0.3074   -0.3294   -0.0033    0.0001
   0.2960   -0.1508   -0.9432   -0.0067    0.0003
   0.3396    0.9395   -0.0436   -0.0129    0.0005
  -0.0034   -0.0121    0.0080   -0.9998   -0.0164
```

```
 -0.0003    -0.0006     0.0004    -0.0164     0.9999
Rk =
-142.7779    -3.0003   -24.3952     0.7548     0.0325
        0  -146.2384   -67.9134     1.6713     0.1029
        0          0   149.9518    -1.8036    -0.1018
        0          0          0   -15.3250    -0.2513
        0          0          0          0    -0.0040
At =
137.4716   -66.3589    50.9233     0.0521     0.0000
-66.3589   -22.5892   140.8997     0.1852     0.0000
 50.9233   140.8997    12.6292    -0.1227    -0.0000
  0.0521     0.1852    -0.1227    34.5027     0.0001
  0.0000     0.0000    -0.0000     0.0001    19.1733
```

请注意:i 表示求第 i 个特征值,tzgk 是矩阵 A 的特征值的近似值,k 是迭代次数,下面的矩阵 B 是 m 阶矩阵 At 的(m-1)阶主子矩阵,即 At 降一阶.

```
i =
     2
tzgk =
19.1733
k =
     6
sk =
19.1773
B =
137.4716   -66.3589    50.9233     0.0521
-66.3589   -22.5892   140.8997     0.1852
 50.9233   140.8997    12.6292    -0.1227
  0.0521     0.1852    -0.1227    34.5027
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量, Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)迭代矩阵:

```
i =
     3
k =
     7
sk =
34.5027
Bk =
102.9689   -66.3589    50.9233     0.0521
```

```
 -66.3589   -57.0919   140.8997    0.1852
  50.9233   140.8997   -21.8735   -0.1227
   0.0521     0.1852    -0.1227         0
Qk =
  -0.7762    -0.5556    -0.2981    0.0003
   0.5002    -0.2548    -0.8276    0.0006
  -0.3839     0.7914    -0.4757   -0.0009
  -0.0004     0.0011     0.0002    1.0000
Rk =
-132.6623   -31.1373    39.3504    0.0993
        0   162.9301   -81.5084   -0.1732
        0          0  -121.3767   -0.1104
        0          0          0    0.0002
At =
 106.7915   112.7868    46.5913   -0.0000
 112.7868   -71.5223   -96.0610    0.0000
  46.5913   -96.0610    92.2421    0.0000
  -0.0000     0.0000     0.0000   34.5029
```

请注意:i 表示求第 i 个特征值,tzgk 是矩阵 A 的特征值的近似值,k 是迭代次数,下面的矩阵 B 是 m 阶矩阵 At 的(m-1)阶主子矩阵,即 At 降一阶.

```
i =
     3
tzgk =
34.5029
k =
     7
sk =
34.5027
B =
106.7915    112.7868    46.5913
112.7868    -71.5223   -96.0610
 46.5913    -96.0610    92.2421
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)迭代矩阵:

```
i =
     4
k =
     8
```

```
sk =
92.2421
Bk =
 14.5494    112.7868    46.5913
112.7868   -163.7644   -96.0610
 46.5913    -96.0610          0
Qk =
 -0.1184      0.9745     0.1906
  0.9177     -0.0341    -0.3957
 -0.3791     -0.2218     0.8984
Rk =
-122.8954    173.3595    82.6437
        0    136.7985    48.6787
        0           0    46.8920
At =
 -83.6398   -144.0011   -17.7774
-144.0011     76.7821   -10.3993
 -17.7774    -10.3993   134.3691
```

请注意：下面的 i 表示求第 i 个特征值，k 是迭代次数，sk 是原点位移量，Bk = Ak - sk * eye(n)，Qk 和 Rk 是 Bk 的 QR 分解，At = Rk * Qk + sk * eye(n) 迭代矩阵：

```
i =
     4
k =
     9
sk =
134.3691
Bk =
-218.0089   -144.0011   -17.7774
-144.0011    -57.5870   -10.3993
-17.7774   -10.3993           0
Qk =
 -0.8325      0.5510    -0.0578
 -0.5499     -0.8345    -0.0357
 -0.0679      0.0020     0.9977
Rk =
261.8782    152.2500    20.5177
       0    -31.3152    -1.1178
       0           0     1.3983
```

```
At =
-168.7515     17.2954    -0.0949
  17.2954    160.4987     0.0029
  -0.0949      0.0029   135.7642
```

请注意:i 表示求第 i 个特征值,tzgk 是矩阵 A 的特征值的近似值,k 是迭代次数,下面的矩阵 B 是 m 阶矩阵 At 的(m-1)阶主子矩阵,即 At 降一阶.

```
i =
      4
tzgk =
135.7642
k =
      9
sk =
134.3691
B =
-168.7515     17.2954
  17.2954    160.4987
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量, Bk = Ak - sk * eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)迭代矩阵:

```
i =
      5
k =
     10
sk =
160.4987
Bk =
-329.2503     17.2954
  17.2954           0
Qk =
  -0.9986      0.0525
   0.0525      0.9986
Rk =
329.7042    -17.2716
       0      0.9073
At =
-169.6576      0.0476
   0.0476    161.4048
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量, Bk = Ak - sk *

eye(n),Qk 和 Rk 是 Bk 的 QR 分解,At = Rk * Qk + sk * eye(n)迭代矩阵:

```
i =
      5
k =
    11
sk =
  161.4048
Bk =
  -331.0623    0.0476
     0.0476         0
Qk =
    -1.0000    0.0001
     0.0001    1.0000
Rk =
   331.0623   -0.0476
          0    0.0000
At =
  -169.6576    0.0000
     0.0000  161.4048
```

请注意:i 表示求第 i 个特征值,tzgk 是矩阵 A 的特征值的近似值,k 是迭代次数,下面的矩阵 B 是 m 阶矩阵 At 的(m-1)阶主子矩阵,即 At 降一阶.

```
i =
      5
tzgk =
  161.4048
k =
    11
sk =
  161.4048
B =
 -169.6576
tzgk =
 -169.6576
```

请注意:n 阶实对称矩阵 A 的全部真特征值 lamoda 和至少含 t 个有效数字的近似特征值 tzg 如下:

```
lamoda =
 -169.6576
  -58.1877
```

```
  19.1733
  34.5029
 135.7642
 161.4048
tzg =
 169.6576
 -58.1877
  19.1733
  34.5029
 135.7642
 161.4048
```

5.6.6 求根位移 QR 方法计算实对称矩阵 A 的特征值及其 MATLAB 程序

为了介绍方便,把用求根位移法选择位移量 s_k 的带原点位移的 QR 方法求 A 的所有近似特征值的方法称作**求根位移 QR 方法**. 这种方法特别适合求实对称矩阵 A 的所有近似特征值. 在这里主要介绍用求根位移 QR 方法求 n 阶实对称矩阵 A 的所有近似特征值的方法和步骤及其 MATLAB 程序.

(一) 求根位移 QR 方法的一般步骤

用求根位移 QR 方法求 n 阶实对称矩阵 A 的所有近似特征值的步骤如下:

步骤 1 用 MATLAB 程序 A1 = house(A) 将 n 阶实对称矩阵 A 化为对称三对角矩阵 A_1;

步骤 2 用求根位移法的 MATLAB 程序选择位移量 s_k.

(1) 根据(5.61)式和(5.63)式编写程序

```
Ak = A1;[n n] - size(Ak); d3 = Ak(n-1,n-1); d4 = Ak(n,n); b3 = Ak(n-1,n);
b4 = Ak(n,n-1); x = solve('x^2 -(d4 + d3) * x + d3 * d4 - b3 * b4 = 0'),
x1 = X(1), x2 = X(2),
```

求出方程(5.63)的根 x_1, x_2;

(2) 用下面的程序选择位移量 s_k,即

```
X1 = abs(x1 - d4); X2 = abs(x2 - d4);
if X1 > X2
    sk = x1;
else
    sk = x2;
end
sk
```

运行后输出结果 s_k.

步骤 3 用 MATLAB 程序计算 $\boldsymbol{A}_k - s_k\boldsymbol{E}$,然后将 $\boldsymbol{A}_k - s_k\boldsymbol{E}$ 进行 QR 分解,即

$$\boldsymbol{A}_k - s_k\boldsymbol{E} = \boldsymbol{Q}_k\boldsymbol{R}_k, k = 1,2,\cdots,n \tag{5.66}$$

可以用下面的 MATLAB 程序完成,即

```
[n n] = size(Ak);
B1 = Ak - sk * eye(n);
[Qk,Rk] = qr(B1);
```

步骤 4 用求根位移法的 MATLAB 程序选择位移量 s_{k+1},构造新矩阵

$$\boldsymbol{A}_{k+1} = \boldsymbol{R}_k\boldsymbol{Q}_k, k = 1,2,\cdots,t_j, \tag{5.67}$$

可以用下面的 MATLAB 程序完成(5.67)的计算,即

```
A2 = R1 * Q1,
```

步骤 5 求特征值.重复执行上述的步骤 2 至步骤 4,直到 $b_{n-1}\approx 0$,可以得到第一个特征值 $\lambda_1 \approx s_1 + s_2 + \cdots + s_{t_1}$;对前 $n-1$ 行和 $n-1$ 列重复执行上述的步骤,直到 $b_{n-2}\approx 0$,可以得到第二个特征值 $\lambda_2 = s_1 + s_2 + \cdots + s_{t_2}$;继续对子矩阵进行迭代,直到得到 $b_2\approx 0$ 和特征值 $\lambda_{n-2} = s_1 + s_2 + \cdots + s_{t_{n-2}}$;最后,用方程(5.63)求解最后两个特征值 λ_{n-1} 和 λ_n.

说明 ①步骤 1 用 MATLAB 程序:Ak = Householdren(A)或 Ak = house(A)运行后,输出 Ak.然后转入②;

②根据步骤 2 编写名为 wyqr1.m 的 M 文件

```
[n n] = size(Ak); d3 = Ak(n-1,n-1); d4 = Ak(n,n);
b3 = Ak(n-1,n); b4 = Ak(n,n-1);
x = solve('x^2 - (d4 + d3) * x + d3 * d4 - b3 * b4 = 0')
```

保存名为 wyqr1.m 的 M 文件后,在 MATLAB 工作窗口输入矩阵 $\boldsymbol{A}$ 和文件名 wyqr1,回车后,即可输出结果 $\boldsymbol{A}_k$ 和方程的根.然后转入③.

③步骤 2 中选择位移量 s_k 和步骤 3,步骤 4 可以编写名为 wyqr2.m 的 M 文件如下:

```
X1 = abs(x1 - d4);
X2 = abs(x2 - d4);
if X1 > X2
    sk = x2;
  else
    sk = x1;
end
sk, [n n] = size(Ak); Bk = Ak - sk * eye(n);
[Qk,Rk] = qr(Bk); Ak = Rk * Qk
```

保存名为 wyqr2.m 的 M 文件后,只需在 MATLAB 工作窗口输入方程的根和文

件名 wyqr2，回车后，即可输出结果 A_t 和 s_k.

例 5.6.6 用求根位移 QR 方法求实对称矩阵 A 全部特征值，精度为 10^{-14}. 并将计算结果与 A 全部真特征值比较. 其中 $A=\begin{pmatrix}4 & 2 & 2 & 1\\ 2 & -3 & 1 & 1\\ 2 & 1 & 3 & 1\\ 1 & 1 & 1 & 2\end{pmatrix}$.

解 (1) 首先求 A 第一个特征值的近似值.

① 用豪斯霍尔德变换约化矩阵 A 为上豪斯霍尔德矩阵.

输入 MATLAB 程序

```
>> A=[4 2 2 1;2 -3 1 1;2 1 3 1;1 1 1 2]; Ak=house(A)
```

运行后屏幕显示结果

```
Ak =
  4.00000000000000  -3.00000000000000                  0                  0
 -3.00000000000000   2.00000000000000   3.16227766016838                  0
                 0   3.16227766016838  -1.40000000000000  -0.20000000000000
                 0                  0  -0.20000000000000   1.40000000000000
```

② 用 wyqr1.m 中的 MATLAB 程序求位移量 s_1 的预选项 x_1 和 x_2.

根据(5.61)式，矩阵 A_1 右下角的四个元为 $d_3=-1.4, d_4=1.4, b_3=b_4=-0.2$，代入(5.63)式，即

$$x^2-(d_4+d_3)x+d_3d_4-b_3b_4=0.$$

输入 MATLAB 程序

```
>> wyqr1
```

回车后屏幕显示结果

```
x1 =1/2*d4+1/2*d3+1/2*(d4^2-2*d3*d4+d3^2+4*b3*b4)^(1/2)
x2 =1/2*d4+1/2*d3-1/2*(d4^2-2*d3*d4+d3^2+4*b3*b4)^(1/2)
```

③ 用 MATLAB 程序 wyqr2.m 进行选择位移量 s_1，计算 A_1-s_1E，并将 A_1-s_1E 进行 QR 分解，使 $A_1-s_1E=Q_1R_1$，然后构造新矩阵 $A_2=R_1Q_1$ 计算.

输入 MATLAB 程序

```
>> x1 =1/2*d4+1/2*d3+1/2*(d4^2-2*d3*d4+d3^2+4*b3*b4)^(1/2)
x2 =1/2*d4+1/2*d3-1/2*(d4^2-2*d3*d4+d3^2+4*b3*b4)^(1/2), wyqr2
```

运行后屏幕显示结果

```
x1 = 1.41421356237309,x2 =  -1.41421356237309
sk = 1.41421356237309
Ak =
 4.40547440373648   2.79049342571929  -0.00000000000000                  0
 2.79049342571929  -4.21662555169097  -0.33010756339024   0.00000000000000
```

```
0   -0.33010756339024     0.21024219752942    -0.03406393608111
0                   0    -0.03406393608111    -0.05594529906731
```

即得到两个根 $x_1 \approx 1.414\ 213\ 562\ 373\ 09$ 和 $x_2 \approx -1.414\ 213\ 562\ 373\ 09$. 选择最接近 $d_4 = 1.4$ 的根 x_1 作为 s_1,即 $s_1 \approx 1.414\ 213\ 562\ 373\ 09$.

$$A_2 = \begin{pmatrix} 4.405\ 474\ 403\ 736\ 48 & 2.790\ 493\ 425\ 719\ 29 & -0.000\ 000\ 000\ 000\ 00 & 0 \\ 2.790\ 493\ 425\ 719\ 29 & -4.216\ 625\ 551\ 690\ 97 & -0.330\ 107\ 563\ 390\ 24 & 0.000\ 000\ 000\ 000\ 00 \\ 0 & -0.330\ 107\ 563\ 390\ 24 & 0.210\ 242\ 197\ 529\ 42 & -0.034\ 063\ 936\ 081\ 11 \\ 0 & 0 & -0.034\ 063\ 936\ 081\ 11 & -0.055\ 945\ 299\ 067\ 31 \end{pmatrix}.$$

④ 用 wyqr1.m 中的 MATLAB 程序求位移量 s_2 的预选项 x_1 和 x_2.

根据(5.61)式,矩阵 A_2 右下角的四个元为

$$d_3 = 0.210\ 242\ 197\ 529\ 42,\ b_3 = -0.034\ 063\ 936\ 081\ 11$$

$$b_4 = -0.034\ 063\ 936\ 081\ 11,\ d_3 = -0.055\ 945\ 299\ 067\ 31,$$

代入(5.63)式,输入程序

```
>> wyqr1
```

运行后根据屏幕显示结果,整理得方程的两个根分别为

```
x1 = 1/2 * d4 + 1/2 * d3 + 1/2 * (d4^2 - 2 * d3 * d4 + d3^2 + 4 * b3 * b4)^(1/2)
x2 = 1/2 * d4 + 1/2 * d3 - 1/2 * (d4^2 - 2 * d3 * d4 + d3^2 + 4 * b3 * b4)^(1/2)
```

⑤ 用 MATLAB 程序 wyqr2.m 进行选择位移量 s_2,计算 $A_2 - s_2E$,并将其进行 QR 分解,使 $A_2 - s_2E = Q_2R_2$,然后计算 $A_3 = R_2Q_2$.

输入 MATLAB 程序

```
>> x1 = 1/2 * d4 + 1/2 * d3 + 1/2 * (d4^2 - 2 * d3 * d4 + d3^2 + 4 * b3 *
b4)^(1/2)
x2 = 1/2 * d4 + 1/2 * d3 - 1/2 * (d4^2 - 2 * d3 * d4 + d3^2 + 4 * b3 * b4)^
(1/2), wyqr2
```

运行后屏幕显示结果

```
x1 = 0.21453220947020, x2 = -0.06023531100809,
sk = -0.06023531100809
Ak =
 4.55257145668091   -2.65725022082954   -0.00000000000000   -0.00000000000000
-2.65725022082954   -4.26046913128464    0.01911255145228   -0.00000000000000
                0    0.01911255145228    0.29171422807115    0.00003195159684
                0                   0    0.00003195159684    0.00027044107257
```

由于两个根 $x_2 \approx -0.060\,235\,311\,008\,09$ 和 $x_1 \approx 0.214\,532\,209\,470\,20$，选择最接近 $d_4 = -0.055\,945\,299\,067\,23$ 的根 x_2 作为 s_2，即 $s_2 = -0.060\,235\,311\,008\,09$.

$$\boldsymbol{A}_3 = \begin{pmatrix} 4.552\,571\,456\,680\,91 & -2.657\,250\,220\,829\,54 & -0.000\,000\,000\,000\,00 & -0.000\,000\,000\,000\,00 \\ -2.657\,250\,220\,829\,54 & -4.260\,469\,131\,284\,64 & 0.019\,112\,551\,452\,28 & -0.000\,000\,000\,000\,00 \\ 0 & 0.019\,112\,551\,452\,28 & 0.291\,714\,228\,071\,15 & 0.000\,031\,951\,596\,84 \\ 0 & 0 & 0.000\,031\,951\,596\,84 & 0.000\,270\,441\,072\,57 \end{pmatrix}.$$

⑥ 同理用上面的 MATLAB 程序选择数列 $s_3 = 2.704\,375\,696\,475\,64\text{e}-004$，计算 $\boldsymbol{A}_3 - s_3\boldsymbol{E}$，并将其进行 QR 分解，使 $\boldsymbol{A}_3 - s_3\boldsymbol{E} = \boldsymbol{Q}_3\boldsymbol{R}_3$. 然后计算 $\boldsymbol{A}_4 = \boldsymbol{R}_3\boldsymbol{Q}_3$，得

$$\boldsymbol{A}_4 = \begin{pmatrix} 4.626\,396\,823\,726\,44 & 2.530\,330\,422\,501\,71 & -0.000\,000\,000\,000\,00 & -0.000\,000\,000\,000\,00 \\ 2.530\,330\,422\,501\,71 & -4.334\,893\,999\,728\,20 & -0.001\,109\,988\,551\,91 & 0.000\,000\,000\,000\,00 \\ 0 & -0.001\,109\,988\,551\,91 & 0.291\,502\,420\,262\,39 & -0.000\,000\,000\,000\,00 \\ 0 & 0 & -0.000\,000\,000\,000\,00 & 0.000\,000\,000\,000\,76 \end{pmatrix}.$$

⑦ 选择数列 $s_4 = 7.552\,847\,236\,524\,94\text{e}-013$，计算 $\boldsymbol{A}_4 - s_4\boldsymbol{E}$，并将其进行 QR 分解，使 $\boldsymbol{A}_4 - s_4\boldsymbol{E} = \boldsymbol{Q}_4\boldsymbol{R}_4$. 然后计算 $\boldsymbol{A}_5 = \boldsymbol{R}_4\boldsymbol{Q}_4$，得

$$\boldsymbol{A}_5 = \begin{pmatrix} 4.693\,517\,581\,969\,55 & -2.407\,608\,460\,616\,26 & -0.000\,000\,000\,000\,00 & -0.000\,000\,000\,000\,00 \\ -2.407\,608\,460\,616\,27 & -4.402\,014\,959\,148\,43 & 0.000\,064\,488\,482\,55 & -0.000\,000\,000\,000\,00 \\ 0 & 0.000\,064\,488\,482\,55 & 0.291\,502\,621\,437\,24 & 0.000\,000\,000\,000\,00 \\ 0 & 0 & 0.000\,000\,000\,000\,00 & 0.000\,000\,000\,000\,00 \end{pmatrix}.$$

因为 $b_4 \approx 0$ 和 $d_4 \approx 0$，所以可以得到一个精度为 10^{-14} 的 $\boldsymbol{A}$ 第一个特征值的近似值：

$$\begin{aligned} \lambda_1 &\approx s_1 + s_2 + s_3 + s_4 \\ &= 7.552\,847\,236\,524\,94\text{e}-013 + 2.704\,375\,696\,475\,64\text{e}-004 - \\ &\quad\ 0.060\,235\,311\,008\,09 + 1.414\,213\,562\,373\,09 \\ &= 1.354\,248\,688\,935\,40 \end{aligned}$$

它与 $\boldsymbol{A}$ 一个特征值 $\lambda_1 = 1.354\,248\,688\,935\,41$ 比较,可见它的精度为 10^{-14}.

(2) 再求 $\boldsymbol{A}$ 第二个特征值的近似值.

将 λ_1 的近似值放到 $\boldsymbol{A}_4$ 的右下角(4,4)的位置,得

$$\boldsymbol{A}_4 = \left(\begin{array}{ccc|c} 4.693\,517\,581\,969\,55 & -2.407\,608\,460\,616\,26 & 0 & 0 \\ -2.407\,608\,460\,616\,27 & -4.402\,014\,959\,148\,43 & 0.000\,064\,488\,482\,55 & 0 \\ 0 & 0.000\,064\,488\,482\,55 & 0.291\,502\,621\,437\,24 & 0 \\ \hline 0 & 0 & 0 & \lambda_1 \end{array}\right).$$

$$= \begin{pmatrix} \boldsymbol{C}_{3\times 3} & \boldsymbol{O}_{3\times 1} \\ \boldsymbol{O}_{1\times 3} & \lambda_1 \end{pmatrix} \tag{5.68}$$

然后,对矩阵 $\boldsymbol{C}_{3\times 3}$ 重复执行上面的过程,即输入 MATLAB 程序

```
>> Ak =[4.69351758196955   -2.40760846061626   -0.00000000000000
   -2.40760846061627   -4.40201495914843    0.00006448848255
                   0    0.00006448848255    0.29150262143724];
wyqr1
```

运行后屏幕显示结果

```
x1 =1/2*d4 +1/2*d3 +1/2*(d4^2 -2*d3*d4 +d3^2 +4*b3*b4)^(1/2)
x2 =1/2*d4 +1/2*d3 -1/2*(d4^2 -2*d3*d4 +d3^2 +4*b3*b4)^(1/2)
```

输入 MATLAB 程序

```
>> x1 =1/2*d4 +1/2*d3 +1/2*(d4^2 -2*d3*d4 +d3^2 +4*b3*b4)^(1/2)
x2 =1/2*d4 +1/2*d3 -1/2*(d4^2 -2*d3*d4 +d3^2 +4*b3*b4)^(1/2),
wyqr2
```

运行后屏幕显示结果

```
s1 =
    0.29150262232331
C2 =
    4.33489425015926    2.53033039508054   -0.00000000000000
    2.53033039508053   -4.62639687267668   -0.00000000000000
                   0    0.00000000000000    0.00000000019413
```

输入 MATLAB 程序

```
>> wyqr1
```

运行后屏幕显示结果

```
x1 =1/2*d4 +1/2*d3 +1/2*(d4^2  2*d3*d4 +d3^2 +4*b3*b4)^(1/2)
x2 =1/2*d4 +1/2*d3 -1/2*(d4^2 -2*d3*d4 +d3^2 +4*b3*b4)^(1/2)
```

输入 MATLAB 程序

```
>> x1 =1/2*d4 +1/2*d3 +1/2*(d4^2 -2*d3*d4 +d3^2 +4*b3*b4)^(1/2)
```

```
x2 =1/2 * d4 +1/2 * d3 -1/2 * (d4^2 -2 * d3 * d4 +d3^2 +4 * b3 * b4)^
(1/2), wyqr2
```

运行后屏幕显示结果

```
x1 =
    -1.941278249262268e-010,x2 = -4.62639687267668,
s2 =
    -1.941278249262268e-010
C3 =
   4.26081409698106   -2.65724252937861    0.00000000000000
  -2.65724252937862   -4.55231671911023   -0.00000000000000
                  0    0.00000000000000   -0.00000000000000
```

因为 $b_4 \approx 0$ 和 $d_4 \approx 0$，所以可以得到一个精度为 10^{-14} 的 $\boldsymbol{A}$ 第二个特征值的近似值：

$$
\begin{aligned}
\lambda_2 &\approx \lambda_1 + s_1 + s_2 \\
&= 1.354\,248\,688\,935\,40 + 0.291\,502\,622\,323\,31 - 1.941\,278\,249\,262\,27\text{e}-010 \\
&= 1.645\,751\,311\,064\,58
\end{aligned}
$$

它与 $\boldsymbol{A}$ 第二个特征值 $\lambda_2 = 1.645\,751\,311\,064\,59$ 比较，可见它的精度为 10^{-14}.

(3) 最后求 $\boldsymbol{A}$ 第三个和第四个特征值的近似值.

将 λ_2 的近似值放到(5.68)的(3,3)的位置，得

$$
\boldsymbol{C}_3 = \left(\begin{array}{cc:cc} 4.260\,814\,096\,981\,06 & -2.657\,242\,529\,378\,61 & 0 & 0 \\ -2.657\,242\,529\,378\,62 & -4.552\,316\,719\,110\,23 & 0 & 0 \\ \hdashline 0 & 0 & \lambda_2 & 0 \\ 0 & 0 & 0 & \lambda_1 \end{array}\right)
= \begin{pmatrix} \boldsymbol{C}_{2\times 2} & \boldsymbol{O}_{2\times 2} \\ \boldsymbol{O}_{2\times 2} & \boldsymbol{\Lambda}_{2\times 2} \end{pmatrix}. \tag{5.69}
$$

最后求矩阵

$$
\boldsymbol{C}_{2\times 2} = \begin{pmatrix} 4.260\,814\,096\,981\,06 & -2.657\,242\,529\,378\,61 \\ -2.657\,242\,529\,378\,62 & -4.552\,316\,719\,110\,23 \end{pmatrix} = \begin{pmatrix} d_3 & b_3 \\ b_4 & d_4 \end{pmatrix}
$$

的特征值，即将 $d_3 = 4.260\,814\,096\,981\,06$， $b_3 = -2.657\,242\,529\,378\,61$，
$b_4 = -2.657\,242\,529\,378\,62$， $d_4 = -4.552\,316\,719\,110\,23$，
代入特征方程(5.63)中求解. 输入 MATLAB 程序

```
>>Ak =[4.26081409698106 -2.65724252937861
-2.65724252937862  -4.55231671911023]; wyqr1
```

运行后屏幕显示结果

```
x1 =1/2 * d4 +1/2 * d3 +1/2 * (d4^2 -2 * d3 * d4 +d3^2 +4 * b3 * b4)^(1/2)
x2 =1/2 * d4 +1/2 * d3 -1/2 * (d4^2 -2 * d3 * d4 +d3^2 +4 * b3 * b4)^(1/2)
```

再输入 MATLAB 程序

```
>> x1 =1/2*d4+1/2*d3+1/2*(d4^2-2*d3*d4+d3^2+4*b3*b4)^(1/2)
   x2 =1/2*d4+1/2*d3-1/2*(d4^2-2*d3*d4+d3^2+4*b3*b4)^(1/2)
```

运行后屏幕显示特征方程(5.63)的根为

```
x1 = 5.00000000000001,x2 = -5.29150262212917
```

矩阵 $\boldsymbol{A}$ 的最后两个近似特征值分别为

$\lambda_3 \approx \lambda_2 + x_1 = 1.645\ 751\ 311\ 064\ 58 + 5.000\ 000\ 000\ 000\ 01 = 6.645\ 751\ 311\ 064\ 59$,

$$\lambda_4 \approx \lambda_2 + x_2 = 1.645\ 751\ 311\ 064\ 58 - 5.291\ 502\ 622\ 129\ 17 = -3.645\ 751\ 311\ 064\ 59.$$

它们与用 MATLAB 程序

```
A=[4 2 2 1;2 -3 1 1;2 1 3 1;1 1 1 2]; eig(A)
```

求出的特征值 $-3.645\ 751\ 311\ 064\ 59$ 和 $6.645\ 751\ 311\ 064\ 59$ 相同.

(二) 用求根位移 QR 方法求实对称矩阵 $\boldsymbol{A}$ 的全部特征值的 MATLAB 程序

如果 $\boldsymbol{A}$ 是 n 阶实对称矩阵,则用带原点位移的 QR 方法计算 $\boldsymbol{A}$ 的所有近似特征值可以用下面的 MATLAB 程序完成.

用求根位移 QR 方法求实对称矩阵 $\boldsymbol{A}$ 的全部特征值的 MATLAB 主程序
输入的量:n 阶实对称矩阵 $\boldsymbol{A}$,计算精度 jd 和最大迭代次数 $max1$;
输出的量:k 是迭代次数,s_k 是中心位移量,$\boldsymbol{A}_k$ 迭代矩阵,矩阵 $\boldsymbol{A}$ 的全部特征值及其相关信息.

根据用求根位移 QR 方法求实对称矩阵 $\boldsymbol{A}$ 的全部特征值的步骤,编写名为 qr8.m 的 MATLAB 主程序如下:

```
function tzg=qr8(A,jd,max1)
[n,n]=size(A); Ak=A; k=0; tzg=zeros(n); state=1;i=1;s0=0;
while((k<=max1)&(state==1)&(n>2))
          bn=abs(Ak(n,n-1));
     if (bn>=jd)
       S=eig(Ak(n-1:n,n-1:n));sk=s0; [j,t]=min([abs(Ak
         (n,n)*[1,1]'-S)]);
       t=t;sk=S(t); Bk= Ak-sk*eye(n);
       [Qk,Rk]=qr(Bk); Ak=Rk*Qk;
       k=k+1;tzgk=sk+s0;s0=tzgk;
       disp('请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是
原点位移量,Ak 迭代矩阵:')
       i,state=1;k,sk,tzgk;Ak,
```

```
            else
                disp('请注意:下面的 i 表示求第 i 个特征值,如果迭代矩阵 Ak 的
阶数 >2,且 m 阶矩阵 Ak 的 m 行第 m-1 列的元近似等于零.则原 n 阶矩阵 A 的第 j 个特征
值 λj = ∑skj,j =1,2,...,n-2;下面的矩阵 Ak 降一阶.')
                i =i +1, k;sk;tzgk,tzg(n,1) =tzgk;Ak;B =Ak(1:n -1,1:n -1);
                Ak =B,n =n -1;state = =1;
            end
        end
                disp('如果迭代矩阵 Ak 的阶数 =2,则原 n 阶矩阵 A 的最后两个特征
                值 λj =λk +xj,k =n -2,j =1,2.')
        for n =2:2
                D =eig(Ak);x1 =D(1),x2 =D(2),tzg1 =tzgk +x1,tzg2 =tzgk +x2,
        end
        tzg(1,1) = tzg1; tzg(2,1) = tzg2;tzg =sort(tzg(:,1));
        disp('请注意:n 阶实对称矩阵 A 的全部真特征值 lamoda 和精度为 jd 的近似
特征值 tzg 如下:')
        lamoda =sort(eig(A))
```

例 5.6.7 用求根位移 QR 方法求实对称矩阵 $\boldsymbol{A}$ 全部特征值,精度为 10^{-4}. 并将计算结果与 $\boldsymbol{A}$ 全部真特征值比较. 其中

$$(1)\ \boldsymbol{A}=\begin{pmatrix}5 & 2 & 2 & 1\\ 2 & -3 & 1 & 1\\ 2 & 1 & 3 & 1\\ 1 & 1 & 1 & 2\end{pmatrix};(2)\ \boldsymbol{A}=\begin{pmatrix}12 & -56 & 3 & -14 & -90 & -41\\ -56 & 71 & 23 & 61 & -9 & -21\\ 3 & 23 & 53 & 12 & -72 & 51\\ -14 & 61 & 12 & 73 & 23 & 21\\ -90 & -9 & -72 & 23 & -34 & -61\\ -41 & -21 & 51 & 21 & -61 & -52\end{pmatrix}.$$

解 (1) 首先把用求根位移 QR 方法求实对称矩阵 $\boldsymbol{A}$ 的全部特征值的 MATLAB 主程序保存为 M 文件,命名为 qr8.m. 然后在工作窗口输入 MATLAB 程序

```
>> A =[5  2  2  1;2  -3  1  1;2  1  3  1;1  1  1  2];
tzg =qr8(A,0.0001,100)
```

运行后屏幕显示结果

```
请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代
矩阵:
    i =
         1
    k =
         1
    sk =
```

```
    1.3820
Ak =
      5.3439   -2.1900    0.3494   -0.0007
     -2.1900   -4.5179   -0.1912    0.0006
      0.3494   -0.1912    0.6437   -0.0019
     -0.0007    0.0006   -0.0019    0.0024
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代矩阵:

```
i =
    1
k =
    2
sk =
    0.0024
Ak =
    5.4899    1.8979    0.0373   -0.0000
    1.8979   -4.6431    0.0211   -0.0000
    0.0373    0.0211    0.6159   -0.0000
   -0.0000   -0.0000   -0.0000   -0.0000
```

请注意:下面的 i 表示求第 i 个特征值,如果迭代矩阵 Ak 的阶数 >2,且 m 阶矩阵 Ak 的 m 行第 m-1 列的元近似等于零.则原 n 阶矩阵 A 的第 j 个特征值 λj = ∑skj,j=1,2,...,n-2;下面的矩阵 Ak 降一阶.

```
i =
    2
tzgk =
    1.3843
Ak =
    5.4899    1.8979    0.0373
    1.8979   -4.6431    0.0211
    0.0373    0.0211    0.6159
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代矩阵:

```
i =
    2
k =
    3
sk =
    0.6160
```

```
Ak =
    4.8235   -2.0282    0.0000
   -2.0282   -5.2085   -0.0000
    0.0000   -0.0000   -0.0004
```

请注意:下面的 i 表示求第 i 个特征值,如果迭代矩阵 Ak 的阶数 >2,且 m 阶矩阵 Ak 的 m 行第 m-1 列的元近似等于零.则原 n 阶矩阵 A 的第 j 个特征值 λj = ∑skj,j =1,2,...,n-2;下面的矩阵 Ak 降一阶.

```
i =
    3
tzgk =
    2.0004
Ak =
    4.8235   -2.0282
   -2.0282   -5.2085
```

如果迭代矩阵 Ak 的阶数 =2,则原 n 阶矩阵 A 的最后两个特征值 λj = λk + xj,k =n-2,j =1,2.

```
x1 =
    5.2180
x2 =
    -5.6030
tzg1 =
    7.2183
tzg2 =
    -3.6027
```

请注意:n 阶实对称矩阵 A 的全部真特征值 lamoda 和精度为 jd 的近似特征值 tzg 如下:

```
lamoda =
   -3.6027
    1.3843
    2.0000
    7.2183
tzg =
   -3.6027
    1.3843
    2.0004
    7.2183
```

(2)首先在工作窗口输入 MATLAB 程序

```
>> A=[12  -56  3  -14  -90  -41;-56  71  23  61  -9  -21;3
```

```
23  53  12  -72  51;-14  61  12  73  23  21;-90  -9  -72  23  -34
-61;-41  -21  51  21  -61  -52];
      Ak = qr8(A,0.0001,100)
```

然后运行后屏幕显示

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代矩阵:

```
i =
   1
k =
   1
sk =
  -104.6604
Ak =
  204.0767  -52.0380   33.2001  -23.0059   42.7318   15.5248
  -52.0380  188.5829   37.4826   39.5606   28.7652   16.0494
   33.2001   37.4826  206.1483    1.2528   10.8059  -25.8849
  -23.0059   39.5606    1.2528  148.3330  -12.3189  -10.6514
   42.7318   28.7652   10.8059  -12.3189  -19.2722  -44.9149
   15.5248   16.0494  -25.8849  -10.6514  -44.9149   23.0934
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代矩阵:

```
i =
   1
k =
   2
sk =
   51.5700
Ak =
  193.6018  -32.5417   32.8853    7.8692  -28.4695    0.5579
  -32.5417  153.8359   57.3434   23.0379  -41.6293    1.0158
   32.8853   57.3434  105.0492  -12.5360   44.8003   -2.1831
   -7.8692   23.0379  -12.5360   77.3704   28.1861   -1.0783
  -28.4695  -41.6293   44.8003   28.1861  -83.1678    2.1298
    0.5579    1.0158   -2.1831   -1.0783    2.1298   -5.1476
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代矩阵:

```
i =
   1
k =
```

```
    3
sk =
    -5.0895
Ak =
  212.2212  -15.9637   20.3171   -0.9641   14.2430    0.0000
  -15.9637  182.5804   44.4902   15.4193   28.2474    0.0001
   20.3171   44.4902   61.8862  -28.6162   69.1715    0.0002
   -0.9641   15.4193  -28.6162   71.1544  -41.6983   -0.0001
   14.2430   28.2474  -69.1715  -41.6983  -55.7556   -0.0001
    0.0000    0.0001   -0.0002   -0.0001   -0.0001   -0.0079
```

请注意：下面的 i 表示求第 i 个特征值，如果迭代矩阵 Ak 的阶数 >2，且 m 阶矩阵 Ak 的 m 行第 m-1 列的元近似等于零．则原 n 阶矩阵 A 的第 j 个特征值 λ j = ∑ skj，j = 1，2，...，n-2；下面的矩阵 Ak 降一阶．

```
i =
    2
tzgk =
    -58.1798
Ak =
  212.2212  -15.9637   20.3171   -0.9641   14.2430
  -15.9637  182.5804   44.4902   15.4193   28.2474
   20.3171   44.4902   61.8862  -28.6162  -69.1715
    0.9641   15.4193   28.6162   71.1544  -41.6983
   14.2430   28.2474  -69.1715  -41.6983  -55.7556
```

请注意：下面的 i 表示求第 i 个特征值，k 是迭代次数，sk 是原点位移量，Ak 迭代矩阵：

```
i =
    2
k =
    4
sk =
    -68.2300
Ak =
  284.0436   -9.6315    7.7123   -2.5643    2.5516
   -9.6315  262.3211   17.7609    3.9340    5.8291
    7.7123   17.7609  155.1351  -10.1496  -26.8565
   -2.5643    3.9340  -10.1496  148.4074  -21.1153
    2.5516    5.8291  -26.8565  -21.1153  -36.6703
```

请注意：下面的 i 表示求第 i 个特征值，k 是迭代次数，sk 是原点位移量，Ak 迭代

矩阵:

```
i =
   2
k =
   5
sk =
   -39.0488
Ak =
  323.9633   -8.3008    4.5758   -1.6668    0.0337
   -8.3008  302.7230   11.0164    1.9231    0.0846
    4.5758   11.0164  196.7647   -6.9213   -0.6039
   -1.6668    1.9231   -6.9213  189.2251   -0.5366
    0.0337    0.0846   -0.6039   -0.5366   -4.1955
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代矩阵:

```
i =
   2
k =
   6
sk =
   -4.1970
Ak =
  328.6763   -7.5149    2.8635   -0.9801    0.0000
   -7.5149  307.2098    7.2600    1.1589    0.0000
    2.8635    7.2600  200.6671   -6.7350   -0.0000
   -0.9801    1.1589   -6.7350  192.9146   -0.0000
    0.0000    0.0000   -0.0000   -0.0000   -0.0020
```

请注意:下面的 i 表示求第 i 个特征值,如果迭代矩阵 Ak 的阶数 >2,且 m 阶矩阵 Ak 的 m 行第 m-1 列的元近似等于零.则原 n 阶矩阵 A 的第 j 个特征值 $\lambda j = \sum skj, j = 1,2,\ldots,n-2$;下面的矩阵 Ak 降一阶.

```
i =
   3
tzgk =
   -169.6556
Ak =
  328.6763   -7.5149    2.8635   -0.9801
   -7.5149  307.2098    7.2600    1.1589
    2.8635    7.2600  200.6671   -6.7350
   -0.9801    1.1589   -6.7350  192.9146
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代矩阵:

```
i =
    3
k =
    7
sk =
    189.0200
Ak =
    140.4576   -6.1713    0.3209   -0.0016
     -6.1713  117.9758    0.7619    0.0020
      0.3209    0.7619   15.1443   -0.1164
     -0.0016    0.0020   -0.1164   -0.1902
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代矩阵:

```
i =
    3
k =
    8
sk =
     -0.1911
Ak =
  141.1467   -5.1611    0.0349   -0.0000
   -5.1611  117.6756    0.1025    0.0000
    0.0349    0.1025   15.3297   -0.0000
   -0.0000    0.0000   -0.0000   -0.0000
```

请注意:下面的 i 表示求第 i 个特征值,如果迭代矩阵 Ak 的阶数 >2,且 m 阶矩阵 Ak 的 m 行第 m-1 列的元近似等于零.则原 n 阶矩阵 A 的第 j 个特征值 $\lambda j=\sum skj, j=1, 2, ..., n-2$;下面的矩阵 Ak 降一阶.

```
i =
    4
tzgk =
    19.1733
Ak =
  141.1467   -5.1611    0.0349
   -5.1611  117.6756    0.1025
    0.0349    0.1025   15.3297
```

请注意:下面的 i 表示求第 i 个特征值,k 是迭代次数,sk 是原点位移量,Ak 迭代

矩阵：

```
i =
    4
k =
    9
sk =
    15.3296
Ak =
  126.2003   -4.1826    0.0000
   -4.1826  101.9627    0.0000
    0.0000    0.0000   -0.0000
```

请注意：下面的 i 表示求第 i 个特征值，如果迭代矩阵 Ak 的阶数 >2，且 m 阶矩阵 Ak 的 m 行第 m-1 列的元近似等于零.则原 n 阶矩阵 A 的第 j 个特征值 λj = ∑skj，j=1，2，...，n-2；下面的矩阵 Ak 降一阶.

```
i =
    5
tzgk =
    34.5030
Ak =
  126.2003   -4.1826
   -4.1826  101.9627
```

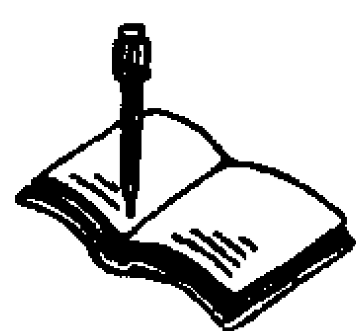
如果迭代矩阵 Ak 的阶数 =2，则原 n 阶矩阵 A 的最后两个特征值 λj = λk + xj，k = n-2，j=1，2.

```
x1 =
    126.9018
x2 =
    101.2613
tzg1 =
    161.4048
tzg2 =
    135.7642
```

请注意：n 阶实对称矩阵 A 的全部真特征值 lamoda 和精度为 jd 的近似特征值 tzg 如下：

```
lamoda =
    -169.6576
    -58.1877
    19.1733
    34.5029
```

```
    135.7642
    161.4048
Ak =
 -169.6556
 -58.1798
  19.1733
  34.5030
  135.7642
  161.4048
```



习 题 5.6

1. 用带原点位移的 QR 方法求实对称矩阵 A 的全部特征值,其中

$$A=\begin{pmatrix} -92 & -5 & 3 & -14 & -90 & -41 \\ -5 & 71 & 23 & 61 & -9 & -21 \\ 3 & 23 & 53 & 12 & -72 & 51 \\ -14 & 61 & 12 & 73 & 23 & 21 \\ -90 & -9 & -72 & 23 & -34 & -61 \\ -41 & -21 & 51 & 21 & -61 & -52 \end{pmatrix}.$$

2. 用初等反射矩阵正交相似约化实矩阵 A 为上豪斯霍尔德矩阵,然后用带原点位移的 QR 方法求实矩阵 A 的全部特征值,其中

$$A=\begin{pmatrix} 67 & -12 & 34 & -12 & 17 & -51 \\ -56 & 7 & 2 & 0 & 32 & -17 \\ 3 & 2 & 5 & 1 & 72 & -63 \\ -1 & 0 & 1 & 12 & 21 & -94 \\ -32 & -78 & -10.2 & 98 & -72 & 11 \\ 31 & -41 & -78 & 37 & -19 & 34 \end{pmatrix}.$$

3. 用求根位移 QR 方法求实对称矩阵 A 的全部特征值,精度为 10^{-4},并将计算结果与 A 的全部真特征值比较,其中

$$(1)\ A=\begin{pmatrix} 5 & 8 & 2 & 1 \\ 8 & -3 & 1 & 1 \\ 2 & 1 & 3 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix};(2)\ A=\begin{pmatrix} 12 & -6 & 3 & -14 & -90 & -41 \\ -6 & 7 & 23 & 61 & -9 & -21 \\ 3 & 23 & 53 & 12 & -72 & 51 \\ -14 & 61 & 12 & 73 & 23 & 21 \\ -90 & -9 & -72 & 23 & -34 & -61 \\ -41 & -21 & 51 & 21 & -61 & -52 \end{pmatrix}.$$

4. 用[Q,R] = qr(A) 和 [Q,R,E] = qr (A) 将矩阵 A 进行正交三角分解，并且比较差异.

$$(1)\ A=\begin{pmatrix}62&-52&34&-12&17&-51\\-56&7&2&0&32&-17\\3&2&5&1&72&-63\\-1&0&1&12&21&-94\\-32&-78&-51/5&98&-72&11\\31&-41&-78&37&-19&34\end{pmatrix};(2)\ A=\begin{pmatrix}21&2&3\\21&2&3\\3&4&5\end{pmatrix}.$$

5. 用[Q,R] = qr(A),[Q1,R1,E1] = qr (A),[Q2,R2] = qr(A,0),[Q3,R3,E3] = qr (A,0),将矩阵 A 进行正交三角分解，并且比较差异.

$$(1)\ A=\begin{pmatrix}7&2&3\\11&2&3\\3&4&5\\7&8&9\\12&3&8\end{pmatrix};(2)\ A=\begin{pmatrix}3&1&5&0&17\\-1&3&2&9&-7\\21&3&8&-6&11\end{pmatrix};(3)\ A=\begin{pmatrix}5&2&3\\1&-5&3\\3&4&5\end{pmatrix}.$$

6. 用 QR 方法解线性方程组 $\begin{cases}3x+4y+5z=1,\\7x+8y+9z=2,\\12x+3y+8z=3,\end{cases}$ 然后再用其他方法验证解的正确性.

5.7 广义特征值问题及其 MATLAB 程序

在工程、物理和化学中常常会遇到一类广义特征值问题，即求复数 λ 和 n 维非零列向量 $X=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$，使得

$$AX=\lambda BX, \tag{5.70}$$

或

$$ABX=\lambda X \tag{5.71}$$

成立，其中 $A=(a_{ij})_{n\times n}$ 是 n 阶实对称矩阵，$B=(b_{ij})_{n\times n}$ 是 n 阶实对称正定矩阵. 在这里把满足(5.70)式的 λ 和 X 分别称为**方阵 A 和 B 的(5.70)型的广义特征值和特征向量**，把满足(5.71)式的 λ 和 X 分别称为**方阵 A 和 B 的(5.71)型的广义特征值和特征向量**. 如果将矩阵 B 作对称三角分解，则这类特征值问题可以转化为一般的对称矩阵的特征值问题.

本节主要介绍(5.70)和(5.71)的广义特征值与特征向量及其直接计算广义特征值问题的 MATLAB 函数的使用方法.

5.7.1 $AX=\lambda BX$ 型的广义特征值和特征向量

(一) 方阵的广义特征值和特征向量

定理 5.30（$AX=\lambda BX$ 型的广义特征值和特征向量） 如果矩阵 $A=(a_{ij})_{n\times n}$是 n 阶实对称矩阵，$B=(b_{ij})_{n\times n}$是 n 阶实对称正定矩阵. 则存在一个非奇异的下三角形矩阵 $L=(l_{ij})_{n\times n}$和 n 维非零列向量 $Y=(y_1,y_2,\cdots,y_n)^{\mathrm{T}}$，使得 $B=LL^{\mathrm{T}}$ 且下列结论成立.

(1) 如果复数 λ 是 C 的特征值，则 λ 也是方阵 A 和 B 的(5.70)型广义特征值. 其中

$$C=L^{-1}A(L^{-1})^{\mathrm{T}}. \tag{5.72}$$

(2) 如果 n 维非零列向量 Y 是 C 的特征向量，则方阵 A 和 B 的(5.70)型广义特征向量为

$$X=(L^{\mathrm{T}})^{-1}Y, \tag{5.73}$$

即

$$x_i=\frac{y_i-\sum\limits_{k=i+1}^{n}l_{ki}x_k}{l_{ii}}\quad(i=1,2,\cdots,n). \tag{5.74}$$

证明 因为 $B=(b_{ij})_{n\times n}$是 n 阶实对称正定矩阵，所以总存在一个非奇异的下三角形矩阵 $L=(l_{ij})_{n\times n}$，使得 $B=LL^{\mathrm{T}}$，从而(5.70)式可写成

$$AX=\lambda LL^{\mathrm{T}}X.$$

上式两端左乘 L^{-1}，得

$$L^{-1}AX=\lambda L^{\mathrm{T}}X,$$

即

$$L^{-1}A(L^{-1})^{\mathrm{T}}L^{\mathrm{T}}X=\lambda L^{\mathrm{T}}X. \tag{5.75}$$

记

$$L^{\mathrm{T}}X=Y, \tag{5.76}$$

则(5.75)式可写成

$$CY=\lambda Y. \tag{5.77}$$

由 A 是 n 阶实对称矩阵和(5.72)式，得 C 也是 n 阶实对称矩阵. 这样，广义特征值问题(5.70)就转化为实对称矩阵的特征值问题. 由于实对称矩阵 C 与 A 相似，且(5.77)式成立，所以 λ 是 C 的特征值，也是方阵 A 和 B 的广义特征值. 但是 Y 是 C 的特征向量，但不是方阵 A 和 B 的广义特征向量. 由(5.76)式得方阵 A 和 B 的广义特征向量为(5.73)式，化简整理得(5.74)式. 证毕

(二) 计算 $AX=\lambda BX$ 型的广义特征值和特征向量的方法和步骤

设 $A=(a_{ij})_{n\times n}$ 是 n 阶实对称矩阵，$B=(b_{ij})_{n\times n}$ 是 n 阶实对称正定矩阵，则计算 $AX=\lambda BX$ 型的广义特征值和特征向量的方法和一般步骤如下：

步骤 1 对 B 进行楚列斯基分解，即求一个非奇异的下三角形矩阵 $L=(l_{ij})_{n\times n}$，使得 $B=LL^{T}$. 因为 B 是对称矩阵，所以只要计算 B 的上三角部分的元即可. 其中 L 可以用下面的公式求得

$$l_{ji}=\begin{cases}\sqrt{b_{ii}-\sum\limits_{k=1}^{i-1}l_{ik}^{2}}, & i=j,\\ \left(b_{ij}-\sum\limits_{k=1}^{i-1}l_{ik}l_{jk}\right)/l_{ii}, & i<j,\\ 0, & i>j.\end{cases}\tag{5.78}$$

步骤 2 计算 $Z=L^{-1}A$. 将(5.72)式改写成 $CL^{T}=L^{-1}A$，得

$$Z=L^{-1}A.\tag{5.79}$$

由(5.79)式计算的矩阵 Z 在一般情况下是非对称矩阵. 但是，因为 A 是对称矩阵，所以 $C=L^{-1}A(L^{-1})^{T}$ 也是对称矩阵，从而只要计算 A 和 C 的上(下)三角部分的元即可. 如果计算 A 的上三角部分的元，则只要计算 Z 的上三角部分的元就够了. 其中 $Z=(z_{ij})_{n\times n}$ 的元可由下面的公式求得

$$z_{ij}=\frac{a_{ij}-\sum\limits_{k=1}^{i-1}l_{ik}z_{kj}}{l_{ii}}\quad(i\leqslant j,i,j=1,2,\cdots,n).\tag{5.80}$$

步骤 3 计算 $C=Z(L^{-1})^{T}$. $C=(c_{ij})_{n\times n}$ 的下三角部分元的计算公式为

$$c_{ji}=\frac{z_{ij}-\sum\limits_{k=1}^{i-1}l_{jk}c_{ik}-\sum\limits_{k=i}^{j-1}l_{jk}c_{ki}}{l_{jj}}\quad(i=1,2,\cdots,n;j=i,i+1,\cdots,n).$$

步骤 4 根据(5.77)式计算 C 的特征值 λ，则 λ 就是方阵 A 和 B 的 $AX=\lambda BX$ 型的广义特征值.

步骤 5 根据(5.77)式计算 C 的对应于特征值 λ 的特征向量 $Y=(y_1,y_2,\cdots,y_n)^{T}$.

步骤 6 根据(5.74)式计算方阵 A 和 B 的 $AX=\lambda BX$ 型的对应于广义特征值 λ 的特征向量 X.

5.7.2 用 MATLAB 计算 $AX=\lambda BX$ 型的广义特征值和特征向量

有许多问题归结为计算 $AX=\lambda BX$ 型的方阵 A 和 B 的广义特征值和特征向

量，而当 A 和 B 是高阶矩阵时，用手工计算相当复杂. 在 MATLAB 函数库中有编好的计算这种类型的广义特征值和特征向量程序，我们只要直接调用这些程序即可. 使用方法见表 5－5.

表 5－5

命令	功能
T = eig (A,B)	输入 n 阶方阵 A 和 B，运行后输出 T 为由方阵 A 和 B 的全部广义特征值构成的列向量
[V,D] = eig (A,B)	输入矩阵 A 和 B，运行后输出矩阵 D 是由 A 和 B 的 $AX=\lambda BX$ 型的全部广义特征值构成的对角矩阵，满矩阵 V 的各列为对应于广义特征值的广义特征向量构成的矩阵，使得 $AV=BVD$
[V,D] = eig (A,B,'chol')	输入对称矩阵 A 和对称的正定矩阵 B，运行后输出矩阵 D 是由 A 和 B 的 $AX=\lambda BX$ 型的全部广义特征值构成的对角矩阵（利用 B 的楚列斯基分解），满矩阵 V 的各列为对应于广义特征值的广义特征向量构成的矩阵，使得 $AV = BVD$
[V,D] = eig(A,B,'qz')	输入矩阵 A 和 B，运行后输出矩阵 D 是由 A 和 B 的全部广义特征值构成的对角矩阵，满矩阵 V 的各列为对应于广义特征值的广义特征向量构成的矩阵

例 5.7.1 选择表 5－5 中两种 MATLAB 程序分别计算方阵 A 和 B 的 $AX=\lambda BX$ 型广义特征值 λ 和它们对应的特征向量 X，比较它们的运算结果. 验证运算结果满足等式 $AV = BVD$. 其中

$$(1)\ A=\begin{pmatrix}1&2&3\\2&3&4\\3&4&5\end{pmatrix},B=\begin{pmatrix}5&6&7\\7&8&9\\8&9&1\end{pmatrix};$$

$$(2)\ A=\begin{pmatrix}1&1&0&-1\\1&1&-1&0\\0&-1&1&1\\-1&0&1&1\end{pmatrix},B=\begin{pmatrix}1&-1&2&1\\-1&3&0&-3\\2&0&9&-6\\1&-3&-6&19\end{pmatrix}.$$

解 (1) 因为 B 不是对称矩阵，所以选择程序 [V,D] = eig (A,B) 和 [V1,D1] = eig (A,B,'qz'). 在 MATLAB 工作窗口输入程序

```
>> A=[1 2 3;2 3 4;3 4 5];B=[5 6 7;7 8 9;8 9 1];
[V,D]=eig (A,B),[V1,D1]=eig (A,B,'qz'),T1=A*V,T2=B*V*D
```

运算后输出结果

```
V =                                    V1 =
-1.0000    0.6552   -0.5000            -1.0000    0.6552   -0.5000
 1.0000   -1.0000    1.0000             1.0000   -1.0000    1.0000
-0.0000    0.0345   -0.5000            -0.0000    0.0345   -0.5000

D =                                    D1 =
   1.0000         0         0             1.0000         0         0
        0    0.5000         0                  0    0.5000         0
        0         0   -0.0000                  0         0   -0.0000
T1 =
1.0e-014 *
   0.1998   -0.1554    0.0222
   0.3331   -0.1554   -0.0444
   0.1998   -0.0888   -0.1276
```

由上面的运算结果可见,程序[V,D] = eig (A,B)和[V1,D1] = eig (A,B,'qz')的运算的结果相同,并且 T1 = A * V - B * V * D 运行结果几乎等于零,即 $\boldsymbol{AV}\approx\boldsymbol{BVD}$. 这说明方阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的 $\boldsymbol{AX}=\lambda\boldsymbol{BX}$ 型广义特征值分别为 $\lambda_1=1.0000$, $\lambda_2=0.5000$, $\lambda_3=-0.0000$,它们对应的特征向量分别为 $\boldsymbol{X}_1=(-1.0000\quad 1.0000\quad -0.0000)^{\mathrm{T}}$, $\boldsymbol{X}_2=(0.6552\ -1.0000\quad 0.0345)^{\mathrm{T}}$, $\boldsymbol{X}_3=(-0.5000\quad 1.0000\quad -0.5000)^{\mathrm{T}}$.

(2) 因为 $\boldsymbol{A}$ 是对称矩阵,且 $\boldsymbol{B}$ 是对称的正定矩阵,所以选择程序[V,D] = eig (A,B,'chol').另外选择程序[V1,D1] = eig (A,B,'qz').在 MATLAB 工作窗口输入程序:

```
>> A=[1 1 0 -1;1 1 -1 0;0 -1 1 1;-1 0 1 1];B=[1 -1 2 1;-1 3 0 3;2 0 9 -6;1 -3 -6 19];
[V,D] = eig (A,B,'chol'), [V1,D1] = eig (A,B,'qz'), T1 = A * V ,T2 = B * V * D
```

运算后输出结果

```
V =
    0.3279    0.2992   -0.0474   -3.6470
   -0.3196   -0.2228    0.4075   -0.8749
   -0.1881    0.1939    0.2588    1.0924
    0.0523   -0.0459    0.2797    0.4086
D =
   -0.1360         0         0         0
         0    0.1416         0         0
         0         0    0.2352         0
         0         0         0   27.5926
```

```
V1 =
    1.0000    1.0000   -1.0000    0.1162
    0.2399   -0.9748    0.7446   -1.0000
   -0.2995   -0.5737   -0.6480   -0.6350
   -0.1120    0.1594    0.1534   -0.6862
D1 =
   27.5926         0         0         0
         0   -0.1360         0         0
         0         0    0.1416         0
         0         0         0    0.2352
T1 =
   -0.0440    0.1223    0.0805   -4.9305
    0.1964   -0.1175    0.1014   -5.6143
    0.1838    0.3708    0.1309    2.3759
   -0.4638   -0.1512    0.5858    5.1480
T2 =
   -0.0440    0.1223    0.0805   -4.9305
    0.1964   -0.1175    0.1014   -5.6143
    0.1838    0.3708    0.1309    2.3759
   -0.4638   -0.1512    0.5858    5.1480
```

由上面的运算结果可见,程序`[V,D] = eig (A,B,'chol')`和`[V1,D1] = eig (A,B,'qz')`的运算的结果 $\boldsymbol{D}$ 和 $\boldsymbol{D}_1$ 相同,并且 $\boldsymbol{T}_1=\boldsymbol{AV}$ 和 $\boldsymbol{T}_2=\boldsymbol{BVD}$ 相等,即 $\boldsymbol{AV}=\boldsymbol{BVD}$. 这说明方阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的 $\boldsymbol{AX}=\lambda\boldsymbol{BX}$ 型广义特征值分别为 $\lambda_1=-0.136\,0$, $\lambda_2=0.141\,6$, $\lambda_3=0.235\,2$, $\lambda_4=27.592\,6$,它们对应的特征向量分别为

$$\boldsymbol{X}_1=(0.327\,9\quad -0.319\,6\quad -0.188\,1\quad 0.052\,3)^{\mathrm{T}},$$

$$\boldsymbol{X}_2=(0.299\,2\quad -0.222\,8\quad 0.193\,9\quad -0.045\,9)^{\mathrm{T}},$$

$$\boldsymbol{X}_3=(-0.047\,4\quad 0.407\,5\quad 0.258\,8\quad 0.279\,7)^{\mathrm{T}},$$

$$\boldsymbol{X}_4=(-3.647\,0\quad -0.874\,9\quad 1.092\,4\quad 0.408\,6)^{\mathrm{T}}.$$

5.7.3 $\boldsymbol{ABX}=\lambda\boldsymbol{X}$ 型的广义特征值和特征向量

(一) 方阵的广义特征值和特征向量

定理 5.31($\boldsymbol{ABX}=\lambda\boldsymbol{X}$ 型的广义特征值和特征向量) 如果矩阵 $\boldsymbol{A}=(a_{ij})_{n\times n}$是 n 阶实对称矩阵,$\boldsymbol{B}=(b_{ij})_{n\times n}$是 n 阶实对称正定矩阵,则存在一个非奇异的下三角形矩阵 $\boldsymbol{L}=(l_{ij})_{n\times n}$和 n 维非零列向量 $\boldsymbol{Y}=(y_1,y_2,\cdots,y_n)^{\mathrm{T}}$,使得 $\boldsymbol{B}=\boldsymbol{LL}^{\mathrm{T}}$ 且下列结论成立.

(1) 如果复数 λ 是 $\boldsymbol{Q}$ 的特征值,则 λ 也是方阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的(5.71)型广义特

征值. 其中

$$Q = L^{T}AL. \tag{5.81}$$

(2) 如果 n 维非零列向量 Y 是 Q 的特征向量,则方阵 A 和 B 的(5.71)型广义特征向量为

$$X = (L^{T})^{-1}Y, \tag{5.82}$$

即

$$x_i = \frac{y_i - \sum_{k=i+1}^{n} l_{ki}x_k}{l_{ii}} \quad (i = 1,2,\cdots,n). \tag{5.83}$$

证明 因为 $B = (b_{ij})_{n\times n}$ 是 n 阶实对称正定矩阵,所以总存在一个非奇异的下三角形矩阵 $L = (l_{ij})_{n\times n}$,使得 $B = LL^{T}$,从而(5.71)式可写成 $ALL^{T}X = \lambda X$.
上式两端左乘 L^{T},得

$$L^{T}ALL^{T}X = \lambda L^{T}X. \tag{5.84}$$

令 $Q = L^{T}AL$,和

$$L^{T}X = Y, \tag{5.85}$$

则(5.84)式可写成

$$QY = \lambda Y. \tag{5.86}$$

由 A 是 n 阶实对称矩阵,得 Q 也是 n 阶实对称矩阵. 这样,广义特征值问题(5.71)就转化为实对称矩阵的特征值问题. 由于实对称矩阵 Q 与 A 相似,且(5.84)式成立,所以 λ 是 Q 的特征值,也是方阵 A 和 B 的广义特征值. 但是 Y 是 Q 的特征向量,但不是方阵 A 和 B 的广义特征向量. 由(5.85)式得,方阵 A 和 B 的广义特征向量为(5.82)式,化简整理得(5.83)式. 证毕

(二) 计算 $ABX = \lambda X$ 型的广义特征值和特征向量的方法和步骤

设 $A = (a_{ij})_{n\times n}$ 是 n 阶实对称矩阵,$B = (b_{ij})_{n\times n}$ 是 n 阶实对称正定矩阵,则计算 $ABX = \lambda X$ 型的广义特征值和特征向量的方法和一般步骤如下:

步骤 1 对 B 进行楚列斯基分解,即求一个非奇异的下三角形矩阵 $L = (l_{ij})_{n\times n}$,使得 $B = LL^{T}$,其中 L 可以用下面的公式求得

$$l_{ji} = \begin{cases} \sqrt{b_{ii} - \sum_{k=1}^{i-1} l_{ik}^{2}}, & i = j, \\ \left(b_{ij} - \sum_{k=1}^{i-1} l_{ik}l_{jk}\right)\Big/ l_{ii}, & i < j, \\ 0, & i > j. \end{cases} \tag{5.87}$$

步骤 2 计算矩阵 $P = AL$.

因为矩阵 Q 是实对称矩阵,只要计算 Q 的下三角部分元. 因此,矩阵 $P = (p_{ij})_{n\times n}$ 是实非对称矩阵,也只要计算矩阵 P 的下三角部分元,这些元可由下面的公式求得

$$p_{ij} = \sum_{k=j}^{i} a_{ki} l_{kj} + \sum_{k=i+1}^{n} a_{ik} l_{kj} (i = 1,2,\cdots,n; j = 1,2,\cdots,i). \qquad (5.88)$$

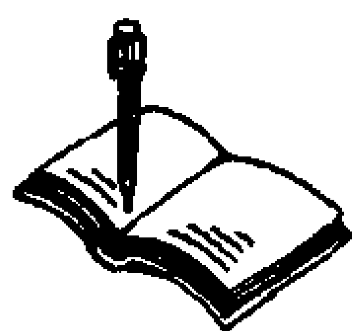
步骤 3 计算 $Q = L^T P$. 根据(5.81)式和 $P = AL$ 可得, $Q = L^T P$. 计算 $Q = (q_{ij})_{n\times n}$ 的下三角部分元的计算公式为

$$q_{ij} = \sum_{k=i}^{n} l_{ki} y_{kj} \quad (i = 1,2,\cdots,n; j = 1,2,\cdots,i). \qquad (5.89)$$

步骤 4 根据(5.86)式计算 Q 的特征值 λ, 则 λ 就是方阵 A 和 B 的 $ABX = \lambda X$ 型的广义特征值.

步骤 5 根据(5.86)式计算矩阵 Q 的对应于特征值 λ 的特征向量 $Y = (y_1, y_2, \cdots, y_n)^T$.

步骤 6 根据(5.82)式计算方阵 A 和 B 的 $AX = \lambda BX$ 型的对应于广义特征值 λ 的特征向量 X.



·习 题 5.7

1. 计算下列各组中矩阵 A 和非对称矩阵 B 的 $AX = \lambda BX$ 型广义特征值 λ 及其对应的特征向量 X, 比较它们的运算结果. 验证运算结果满足等式 $AV = BVD$. 其中

(1) $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{pmatrix}$; $B = \begin{pmatrix} 5 & 6 & 7 \\ 7 & 8 & 9 \\ 8 & 9 & 1 \end{pmatrix}$;

(2) $A = \begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 1 \\ -1 & 0 & 1 & 1 \end{pmatrix}$; $B = \begin{pmatrix} 1 & -1 & 2 & 1 \\ -1 & 3 & 0 & -3 \\ 2 & 0 & 9 & -6 \\ 1 & -3 & -6 & 19 \end{pmatrix}$.

2. 计算上题各组中矩阵 A 和 B 的 $ABX = \lambda X$ 型广义特征值 λ 及其对应的特征向量 X, 比较它们的运算结果.

第六章 函数的插值方法

在科学研究和工程以及日常生活中，常常会遇到计算函数值等一类问题，然而函数关系往往是很复杂的，甚至没有明显的解析表达式. 例如，我们在数学用表上要查 sin 35°16′，可是表上每 10′才有一个函数值，四位数学用表里只有 sin 35°10′ = 0.576 0 和 sin 35°20′ =0.578 3，没有 sin 35°16′的值. 我们可以认为在 35°10′到 35°20′这样小的范围内，正弦函数可以近似为线性函数，于是很容易地得到 sin 35°16′ =0.576 0 + (0.578 3 −0.576 0) ×0.6 =0.577 4. 我们用的这种办法是一种插值方法——分段线性插值. 实际上，插值可以理解为，要根据一个用表格表示的函数，计算表中没有的函数值. 表中有的，如 (sin 35°10′，0.576 0)，(sin 35°20′，0.578 3) 称为**节点**；要计算的，如 35°16′，称为**插值点**，结果 0.577 4 为**插值**. 我们作的线性函数为**插值函数**，插值函数所表示的直线当然要通过节点.

插值在数学发展史上是个古老问题. 它是和拉格朗日 (Lagrange)、牛顿 (Newton)、高斯 (Gauss) 等著名数学家的名字连在一起的. 它最初来源于天体计算——由若干观测值 (即节点) 计算任意时刻星球的位置 (即插值点和插值)——的需要. 现在，虽然人们已很少需要用它从函数表计算函数值了，但是插值仍然在诸如机械加工等工程技术和数据处理等科学研究中有着许多直接的应用. 另一方面，插值又是数值微分、数值积分、常微分方程数值解等数值计算的基础. 本章介绍插值问题的提法和常用的插值. 例如，拉格朗日多项式插值、牛顿插值、埃尔米特 (Hermite) 插值、高元插值、分段插值和三次样条插值及其 MATLAB程序等.

6.1 插值问题及其误差

本节介绍插值问题的有关概念，求解的基本思路，插值多项式存在和唯一性，插值余项和误差限. 最后介绍 MATLAB 命令 `poly(A)`，`conv(A,B)` 和 `deconv(A,B)` 等的功能和调用方法.

6.1.1 插值多项式的唯一性和插值余项

插值问题的提法是，已知 $n+1$ 个节点 (x_j, y_j)，$j=0,1,\cdots,n$，其中 x_j 互不相同，不妨设 $a \leqslant x_0 < x_1 < \cdots < x_n \leqslant b$，求任一插值点 x^* ($\neq x_j$) 处的插值 y^*. 其中

$y_j(j=0,1,2,\cdots,n)$可以看成是由某个函数$y=g(x)$产生的,$g(x)$的解析表达式可能十分复杂,或不存在封闭形式,也可以未知.

求解的基本思路是,构造一个相对简单的函数$y=f(x)$,使$f(x)$通过全部节点,即$f(x_j)=y_j(j=0,1,\cdots,n)$,再用$f(x)$计算插值,即$y^*=f(x^*)$.

从理论和计算角度看,多项式是最简单的函数,设$P_n(x)$是n次多项式,记作

$$P_n(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0. \tag{6.1}$$

对于节点(x_j,y_j)应有

$$P_n(x_j)=y_j,\quad j=0,1,\cdots,n \tag{6.2}$$

为了确定插值多项式$P_n(x)$中的系数$a_n,a_{n-1},\cdots,a_0$,将(6.2)代入(6.1),有

$$\begin{cases} a_nx_0^n+a_{n-1}x_0^{n-1}+\cdots+a_1x_0+a_0=y_0,\\ \cdots\cdots\cdots\cdots \\ a_nx_n^n+a_{n-1}x_n^{n-1}+\cdots+a_1x_n+a_0=y_n. \end{cases} \tag{6.3}$$

记$\boldsymbol{X}=\begin{pmatrix} x_0^n & x_0^{n-1} & \cdots & 1\\ \vdots & \vdots & & \vdots \\ x_n^n & x_n^{n-1} & \cdots & 1\end{pmatrix}$, $\boldsymbol{A}=(a_n,a_{n-1},\cdots,a_0)^{\mathrm{T}}$, $\boldsymbol{Y}=(y_0,y_1,\cdots,y_n)^{\mathrm{T}}$.

方程组(6.3)简写作

$$\boldsymbol{XA}=\boldsymbol{Y}, \tag{6.4}$$

其中 det $\boldsymbol{X}$ 是范德蒙德行列式. 因为$n+1$个节点(x_j,y_j) $(j=0,1,\cdots,n)$的横坐标x_j互不相同,根据行列式性质可得

$$\det \boldsymbol{X}=\prod_{0\leqslant j<k\leqslant n}(x_k-x_j)\neq 0.$$

于是矩阵方程(6.4)中$\boldsymbol{A}$有唯一解,即根据$n+1$个节点可以确定唯一的n次插值多项式.

定理 6.1 设$n+1$个节点(x_j,y_j) $(j=0,1,\cdots,n)$的横坐标满足$a\leqslant x_0<x_1<\cdots<x_n\leqslant b$,且$\omega_n(x)=(x-x_n)\cdots(x-x_1)(x-x_0)$,则

(1) 满足插值条件

$$f(x_j)=y_j,\quad j=0,1,\cdots,n$$

的n次插值多项式(6.1)存在且唯一.

(2) n次插值多项式(6.1)对于任意$x\in(a,b)$,有插值余项

$$R_n(x)=f(x)-P_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_n(x),\xi\in(a,b). \tag{6.5}$$

(3) 如果存在一个正数M_{n+1},使得$|f^{(n+1)}(\xi)|\leqslant M_{n+1}$,则

$$|R_n(x)|\leqslant\frac{M_{n+1}}{(n+1)!}|\omega_n(x)|. \tag{6.6}$$

证明 (2) 设x是$[a,b]$中任一固定点,则有节点和非节点两种情况.

当 x 是节点 $x_i(i=1,2,\cdots,n)$ 时，

$$\frac{f^{(n+1)}(\xi)}{(n+1)!}(x_i-x_n)\cdots(x_i-x_1)(x_i-x_0)=0,$$

故 $R(x_i)=f(x_i)-P_n(x_i)=0$.

当 x 不是节点 $x_i(i=1,2,\cdots,n)$ 时，作辅助函数

$$\varphi(t)=f(t)-P_n(t)-\frac{(t-x_n)\cdots(t-x_1)(t-x_0)}{(x-x_n)\cdots(x-x_1)(x-x_0)}[f(x)-P_n(x)].$$

由 $\omega_n(t)=(t-x_n)\cdots(t-x_1)(t-x_0)$，得

$$\varphi(t)=f(t)-P_n(t)-\frac{\omega_n(t)}{\omega_n(x)}[f(x)-P_n(x)]. \tag{6.7}$$

当 $t=x,x_0,x_1,\cdots,x_n$ 时，$\varphi(t)=0$，所以 $\varphi(t)$ 在 $[a,b]$ 上有 $n+2$ 个互异的零点. 根据罗尔(Rolle)定理知，$\varphi'(t)$ 分别在连续函数 $\varphi(t)$ 每两个相邻的零点之间至少有一个零点 $\xi_i^{(1)}(i=1,2,\cdots,n,n+1)$，使得 $\varphi'(\xi_i^{(1)})=0$. 不妨设 $\xi_i^{(1)}<\xi_{i+1}^{(1)}$ $(i=1,2,\cdots,n)$，对 $\varphi'(t)$ 在 $[\xi_i^{(1)},\xi_{i+1}^{(1)}]$ $(i=1,2,\cdots,n)$ 上应用罗尔定理，知在 $(\xi_i^{(1)},\xi_{i+1}^{(1)})$ $(i=1,2,\cdots,n)$ 内至少存在一点 $\xi_i^{(2)}(i=1,2,\cdots,n)$，使得 $\varphi''(\xi_i^{(2)})=0$. 依此类推可得，$\varphi^{(n)}(t)$ 在 $(\xi_1^{(n)},\xi^{(n)})\in(a,b)$ 内至少存在一点 ξ，使得 $\varphi^{(n+1)}(\xi)=0$. 又因为 $P_n(x)$ 是 n 次多项式，故 $P_n^{(n+1)}(t)=0$.

将(6.7)式求 $t=\xi$ 处的 $n+1$ 阶导数，得

$$\varphi^{(n+1)}(\xi)=f^{(n+1)}(\xi)-\frac{(n+1)!}{\omega_n(x)}[f(x)-P_n(x)]=0.$$

故，插值误差

$$R_n(x)=f(x)-P_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_n(x),\ \xi\in(a,b).$$

从而

$$f(x)=P_n(x)+\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_n(x),\ \xi\in(a,b).$$

证毕

n 次插值多项式 $P_n(x)$ 的唯一性，保证我们无论用什么方法获得的相同条件下的插值多项式，可能形式不同，但都是同一个多项式. 另外，可以用(6.5)式或(6.6)式估计用 $P_n(x)$ 近似表示 $f(x)$ 时的误差.

为此我们首先介绍在 MATLAB 函数库中与插值有关的 MATLAB 函数的功能.

6.1.2 与插值有关的 MATLAB 函数

在这里我们把多项式 $f(x)$ 按降幂排列的系数所构成的向量 V 称为多项式 $f(x)$ 的系数向量. 下面介绍 MATLAB 函数 poly2sym，poly，conv 和 deconv 等的功能和调用方法.

(一) poly2sym 函数

poly2sym 的功能是把多项式的系数向量转化为符号多项式,其调用格式如下:

调用格式一: poly2sym (C)

如果输入 $n+1$ 维向量 $\boldsymbol{C}$,其中 $\boldsymbol{C}$ 是多项式 $f(x)$ 的系数向量,则运行后输出 $f(x)$ 是以 x 为自变量的 n 次多项式.

例如,输入

```
>> f =poly2sym([12 0  -22  -53])
```

则运行后输出结果

```
f =
   12 * x^3 -22 * x -53
```

调用格式二: f1 =poly2sym(C,'V')或 f2 =poly2sym(C, sym ('V'))

poly2sym(C,'V')和 poly2sym(C,sym('V'))两者都用于返回以 V 为自变量,系数向量 $\boldsymbol{C}$ 的符号多项式.

例如,输入

```
>> f1 =poly2sym([12 0  -22  -53],'t'),
   f2 =poly2sym([12 0  -22  -53],sym('t')),
```

则运行后输出相同的结果

```
f1 =                          f2 =
   12 * t^3 -22 * t -53          12 * t^3 -22 * t -53
```

(二) polyval 函数

polyval 函数的主要功能是估计多项式的值.

调用格式: Y =polyval(P,X)

(1) 如果输入数 X 和 $n+1$ 维向量 $\boldsymbol{P}$,其中 $\boldsymbol{P}$ 是 n 次多项式 $f(x)$ 的系数向量,则运行后输出 $f(x)$ 在 $x=X$ 的函数值 $Y=f(X)$,即

$$Y=\boldsymbol{P}(1)X^{n}+\boldsymbol{P}(2)X^{n-1}+\cdots+\boldsymbol{P}(N)X+\boldsymbol{P}(N+1).$$

例如,输入程序

```
>> P =[12 0  -22  -53]; Y = polyval(P,1.5),
```

则运行后输出函数值 $f(1.5)$ 为

```
Y =
    -45.500 0;
```

(2) 如果输入的 $\boldsymbol{X}$ 是矩阵或者是向量,运行后输出在 $\boldsymbol{X}$ 所有元素处的多项式的估计值的矩阵或者是向量 $\boldsymbol{Y}$;

例如,输入程序

```
>> X =[1,2,3;4,5,6]; P =[12 0  -22  -53];Y =polyval(P,X)
```

则运行后输出函数值的矩阵为

```
Y =
    -63      -1      205
    627     1337     2407
```

(三) poly 函数

poly 命令的主要功能是把根转换成多项式的系数向量.

调用格式:Y = poly (V)

(1) 如果输入的 V 是 n 维向量,V 的元素是方程 $f(x)=0$ 的根,则运行后输出 $n+1$ 维的行向量 P,其中

$$f(x)=P(1)X^n+P(2)X^{n-1}+\cdots+P(N)X+P(N+1).$$

(2) 如果输入的 V 是 n 阶方阵,运行后输出 $n+1$ 维的行向量 P,P 是方阵 V 的特征多项式的系数向量.

例 6.1.1 求 3 阶方阵 A 的特征多项式. 其中 $A=\begin{pmatrix}2&0&1\\0&3&2\\0&4&7\end{pmatrix}$.

解 在 MATLAB 工作窗口输入

```
>> A = [2 0 1;0 3 2;0 4 7]; P = poly(A), T = poly2sym (P)
```

运行后输出 A 的特征多项式 T 及其系数向量 P 为

```
P =
    1.0000   -12.0000   33.0000   -26.0000
T =
    x^3 -12 * x^2 +33 * x -26
```

(四) conv 函数

conv 的功能是计算卷积和多项式的积.

调用格式:C = conv(A,B)

如果输入 n 维向量 A 和 m 维向量 B,其中 A 是多项式 $f(x)$ 的系数向量,B 是多项式 $g(x)$ 的系数向量,则运行后输出 $n+m-1$ 维的行向量 C ,其中 C 是多项式 $f(x)$ 与 $g(x)$ 的积 $f(x)\cdot g(x)$ 的系数向量.

例 6.1.2 求三个一次多项式 $g(x)$,$h(x)$ 和 $f(x)$ 的积 $f(x)\cdot g(x)\cdot h(x)$,它们的零点分别依次为 0.4,0.8,1.2.

解 我们可以用两种 MATLAB 程序求之.

方法 1 如输入 MATLAB 程序

```
>> X1 = [0.4,0.8,1.2]; l1 = poly(X1), L1 = poly2sym (l1)
```

运行后输出结果为

```
l1 =
    1.0000   -2.4000   1.7600   -0.3840
L1 =
```

```
x^3 -12/5*x^2 +44/25*x -48/125
```

方法 2 如输入 MATLAB 程序

```
>> P1 =poly(0.4);P2 =poly(0.8);P3 =poly(1.2);
C =conv (conv (P1, P2), P3) , L1 =poly2sym (C)
```

运行后输出的结果与方法 1 相同.

(五) deconv 函数

deconv 的功能是计算逆卷积和多项式的除法、商和余式.

调用格式:[Q,R] =deconv (B,A)

如果输入 n 维向量 $\boldsymbol{A}$ 和 m 维向量 $\boldsymbol{B}$,其中 $\boldsymbol{A}$ 是多项式 $f(x)$ 的系数向量,$\boldsymbol{B}$ 是多项式 $g(x)$ 的系数向量,则运行后输出 $g(x)$ 被 $f(x)$ 除所得的商的系数向量 $\boldsymbol{Q}$ 和余式的系数向量 $\boldsymbol{R}$,使得 $\boldsymbol{B}=\mathrm{conv}(\boldsymbol{A},\boldsymbol{Q})+\boldsymbol{R}$.

例 6.1.3 求多项式 $g(x)=15x^5-32x^2+4x-89$ 被 $f(x)=16x^2-3x+7$ 除后的结果.

解 如果输入 6 维向量 $\boldsymbol{B}=(15, 0, 0, -32, 4, -89)$ 和 3 维向量 $\boldsymbol{A}=(16, -3, 7)$,其中 $\boldsymbol{A}$ 是多项式 $f(x)$ 的系数向量,$\boldsymbol{B}$ 是多项式 $g(x)$ 的系数向量,即输入程序

```
>> B =[15, 0, 0 , -32, 4, -89];A =[16, -3, 7];
[Q,R] =deconv (B,A),q =poly2sym (Q),r =poly2sym (R)
```

则运行后输出 $g(x)$ 被 $f(x)$ 除所得的商的系数向量 $\boldsymbol{Q}$ 和余式的系数向量 $\boldsymbol{R}$ 及其对应的多项式如下

```
Q =
    0.9375   0.1758   -0.3772   -2.1476
R =
    0    0    0    0    0.1975    -73.9666
q =
    15/16*x^3 +45/256*x^2 -1545/4096*x -140747/65536
r =
    12943/65536*x -4847475/65536
```

使得 $\boldsymbol{B}=\mathrm{conv}(\boldsymbol{A},\boldsymbol{Q})+\boldsymbol{R}$.

(六) roots(poly(1:n))命令

调用格式:roots(poly(1:n))

如果输入的 1:n 是1到 n 的等差数列($n\leqslant 20$),且此等差数列的元素是方程 $f(x)=0$ 的全部根,则运行后输出 n 维的行向量 $\boldsymbol{P}$,$\boldsymbol{P}$ 的元素是多项式 $f(x)$ 的零点,即方程 $f(x)=0$ 的全部近似根.

例 6.1.4 试举两个实例验证:当 $n\leqslant 20$ 时,运行命令 P = roots(poly(1:n))后输出以 n 维行向量 $\boldsymbol{X}=(1,2,\cdots,n)$ 的元素为方程 $f(x)=0$ 的全部根

的近似根. 当 $n>20$ 时,则不一定.

解 分别取 $n=20, n=21$. 然后在 MATLAB 工作窗口输入程序

```
>> f1 =poly(1:20);
f =poly2sym (f1),
P =roots(f1)',
P1 =roots(poly(1:21))'
```

运行后输出以 1 到 20 的正整数为方程 $f(x)=0$ 的全部根的函数 f 和全部根的近似根 P 及 P1 =roots(poly(1:21))的结果 P_1 分别如下

```
f =
x^20 -210 * x^19 +20615 * x^18 -1256850 * x^17 +53327946 * x^16 -
1672280820 * x ^ 15 + 40171771630 * x ^ 14 - 756111184500 * x ^ 13 +
11310276995381 * x^12 -135585182899530 * x^11 +1307535010540395 * x^10 -
10142299865511450 * x^9 +63030812099294896 * x^8 -311333643161390656 * x
^ 7 + 1206647803780373248 * x ^ 6 - 3599979517947607040 * x ^ 5 +
8037811822645052416 * x ^ 4 - 12870931245150988288 * x ^ 3 +
13803759753640704000 * x ^ 2 - 8752948036761600000 * x +
2432902008176640000
P =
    Columns 1 through 6
    20.0003   18.9970   18.0118   16.9695   16.0509   14.9319
    Columns 7 through 12
    14.0684   12.9472   12.0345   10.9836   10.0063   8.9983
    Columns 13 through 18
    8.0003   7.0000   6.0000   5.0000   4.0000   3.0000
    Columns 19 through 20
    2.0000   1.0000
P1 =
    Columns 1 through 6
     20.9978   20.0216   18.8721   18.2697   16.5546 - 0.4682i
16.5546 +0.4682i
    Columns 7 through 12
    14.4445 -0.4643i   14.4445 +0.4643i   12.7349   12.1230
10.9837   9.9971
    Columns 13 through 18
    9.0026   7.9993   7.0001   6.0000   5.0000   4.0000
    Columns 19 through 21
    3.0000   2.0000   1.0000
```

由此可见,当 $n=20$ 时,运行命令 P =roots(poly(1:n))后输出的向量 $\boldsymbol{P}$

是以 20 维行向量 $\boldsymbol{X}=(1,2,\cdots,20)$ 的元素为方程 $f(x)=0$ 的全部根的近似根. 但是,当 $n=21$ 时,运行命令 `P1 = roots(poly(1:n))` 后输出的向量 $\boldsymbol{P}_1$ 不全是以 21 维行向量 $\boldsymbol{X}=(1,2,\cdots,21)$ 的元素为方程 $f(x)=0$ 的全部根的近似根.

(七) `det(a * eye(size (A)) - A)`**命令**

调用格式:`b = det(a * eye(size (A)) - A)`

如果输入的是 n 阶方阵 $\boldsymbol{A}$ 和数 a,运行后输出的值 b. 如果 b 是零,则数 a 是 $\boldsymbol{A}$ 的特征值;如果 b 不是零,则数 a 不是 $\boldsymbol{A}$ 的特征值.

例 6.1.5 判断数 $a=2$ 和 $b=1$ 是否是例 6.1.1 中矩阵 $\boldsymbol{A}$ 的特征值.

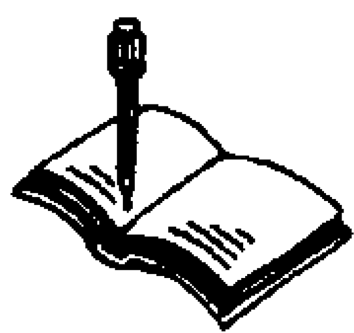
解 在 MATLAB 工作窗口输入程序

```
>> A =[2 0 1;0 3 2;0 4 7];
   a =2;a1 =det(a * eye(size(A)) - A), b =1;b1 =det(b * eye(size(A))
 - A)
```

运行后输出结果为

```
a1 =          b1 =
    0              -4
```

所以 $a=2$ 是矩阵 $\boldsymbol{A}$ 的特征值,而 $b=1$ 不是 $\boldsymbol{A}$ 的特征值.



习 题 6.1

1. 求三个一次多项式 $g(x)$, $h(x)$ 和 $f(x)$ 的积 $f(x)\cdot g(x)\cdot h(x)$,它们的零点分别为 0.2,0.5,1.3.

2. 求多项式 $g(x)=5x^5-2x^2+4x-9$ 被 $f(x)=6x^2-3x+7$ 相除后的结果.

6.2 拉格朗日(Lagrange)插值及其 MATLAB 程序

先考察几种简单的情形,然后推广到一般的情形.

6.2.1 线性插值及其 MATLAB 程序

问题 设函数 $f(x)$ 在 $[x_0,x_1]$ 上具有二阶连续导数,且满足条件

$$f(x_0)=y_0,\quad f(x_1)=y_1, \tag{6.8}$$

求作插值多项式 $L_1(x)$,使满足条件 $L_1(x_0)=y_0$, $L_1(x_1)=y_1$ 并根据罗尔定理确定其误差.

两点只能唯一地确定一个一次多项式,即线性插值. 由直线方程的两点式有

$$L_1(x)-y_0=\frac{y_1-y_0}{x_1-x_0}(x-x_0).$$

经整理得

$$L_1(x)=\frac{x-x_1}{x_0-x_1}y_0+\frac{x-x_0}{x_1-x_0}y_1. \tag{6.9}$$

取

$$l_0=\frac{x-x_1}{x_0-x_1},\quad l_1=\frac{x-x_0}{x_1-x_0}, \tag{6.10}$$

则(6.9)可以表示为

$$L_1(x)=y_0l_0+y_1l_1,$$

且满足　　$l_0(x_0)=1,\quad l_1(x_1)=1,\quad l_0(x_1)=0,\quad l_1(x_0)=0,$

即

$$l_i(x_j)=\begin{cases}1,i=j,\\0,i\neq j\end{cases}\quad(i,j=0,1).$$

从而(6.9)式满足(6.8),即为所求.取误差

$$R_1(x)=f(x)-L_1(x)=g(x)(x-x_1)(x-x_0).$$

当取定 $t\in(x_0,x_1)$ 时,函数

$$\varphi(x)=f(x)-L_1(x)-g(x)(x-x_1)(x-x_0)$$

有三个零点 x_0,x_1,t. 不失一般性,假设 $x_0<t<x_1$,分别在$[x_0,t]$,$[t,x_1]$上应用罗尔定理,知在(x_0,t),(t,x_1)内至少存在一点 ξ_1,ξ_2 使得 $\varphi'(\xi_1)=0,\varphi'(\xi_2)=0$,在$[\xi_1,\xi_2]$上再应用罗尔定理,得在$(\xi_1,\xi_2)$内至少存在一点 ξ,使得

$$\varphi''(\xi)=f''(\xi)-2!\ g(x)=0.$$

从而有 $g(x)=\frac{1}{2!}f''(\xi)$. 故插值误差

$$R_1(x)=\frac{1}{2!}f''(\xi)(x-x_1)(x-x_0),\quad \xi\in(x_0,x_1). \tag{6.11}$$

因此,有

$$f(x)=L_1(x)+\frac{1}{2!}f''(\xi)(x-x_1)(x-x_0).$$

当$f(x)\approx L_1(x)$时,可以用(6.11)式估计误差.(6.9)式称**线性插值函数**,也称**两点一次插值函数**,(6.10)式称**线性插值基函数**,用 $L_1(x)$计算插值称**线性插值**.

例 6.2.1　已知函数$f(x)$在$[1,3]$上具有二阶连续导数,$|f''(x)|\leqslant 5$,且满足条件$f(1)=1$, $f(3)=2$. 求线性插值多项式和函数值$f(1.5)$,并估计其误差.

解　下面分别用手工计算和 MATLAB 程序计算.

方法 1(手工计算)　将 $x_0=1,x_1=3,y_0=1,y_1=2$ 代入(6.9)式,得

$$L_1(x)=\frac{x-3}{1-3}\cdot 1+\frac{x-1}{3-1}\cdot 2=\frac{1}{2}(x+1),$$

$$f(1.5)\approx L_1(1.5)=\frac{1}{2}(1.5+1)=1.25.$$

根据(6.11)式可以估计其误差为

$$|R_1(1.5)| = \frac{1}{2!}|f''(\xi)||(1.5-1)(1.5-3)| \leq 1.875, \quad \xi \in (1,3).$$

方法 2(MATLAB 程序计算) 输入程序

```
>> X=[1,3];Y=[1,2]; l01 = poly(X(2))/( X(1) - X(2)),
   l11 = poly(X(1))/( X(2) - X(1)),
   l0 =poly2sym (l01),l1 =poly2sym (l11),
   P = l01* Y(1) + l11* Y(2),
   L =poly2sym (P)',x =1.5; Y = polyval(P,x)
```

运行后输出基函数 l_0 和 l_1 及其插值多项式的系数向量 P(略)、插值多项式 L 和插值 Y 为

```
l0 =                l1 =              L =               Y =
  -1/2*x+3/2        1/2*x-1/2         1/2*x+1/2           1.2500
```

输入程序

```
>> M=5;R1 =M*abs((x-X(1))* (x-X(2)))/2
```

运行后输出误差限为

```
R1 =
        1.8750
```

例 6.2.2 求函数 $f(x)=\mathrm{e}^{-x}$在$[0,1]$上线性插值多项式,并估计其误差.

解 下面分别用手工计算和 MATLAB 程序计算.

方法 1(手工计算) 将 $x_0=0, x_1=1, y_0=f(0)=1, y_1=f(1)=\mathrm{e}^{-1}$代入(6.9)式,得

$$L_1(x) = \frac{x-1}{0-1}\cdot 1 + \frac{x-0}{1-0}\cdot \mathrm{e}^{-1} = 1+(\mathrm{e}^{-1}-1)x,$$

即

$$\mathrm{e}^{-x} \approx 1 - 0.632\,120\,56x.$$

根据(6.11)式可以估计其误差为

$$R_1(x) = \frac{1}{2!}f''(\xi)(x-x_1)(x-x_0) = \frac{1}{2}\mathrm{e}^{-\xi}x(x-1), \quad \xi \in (0,1).$$

因为 $\xi \in (0,1)$, $|\mathrm{e}^{-\xi}|<1$, $\max\limits_{0\leq x\leq 1}|x(x-1)|=0.25$,所以

$$|R_1(x)| \leq \frac{1}{2}|\mathrm{e}^{-\xi}||x(x-1)| \leq 0.125.$$

方法 2(MATLAB 程序计算) 输入程序

```
>> X=[0,1]; Y =exp( -X),
   l01 =poly(X(2))/( X(1) - X(2)),
   l11 = poly(X(1))/( X(2) - X(1)), l0 =poly2sym (l01),
   l1 =poly2sym (l11), P = l01* Y(1) + l11* Y(2), L =poly2sym (P),
```

运行后输出基函数 l_0 和 l_1 及其插值多项式的系数向量 $\boldsymbol{P}$、插值多项式 $\boldsymbol{L}$ 为

```
10 =            11 =            P =

  -x+1             x               -0.6321    1.0000

L =

  -1423408956596761/2251799813685248*x+1
```

输入程序

```
>> M=1;x=0:0.001:1; R1=M*max(abs((x-X(1)).*(x-X(2))))./2
```

运行后输出误差限为

```
R1 =

  0.1250.
```

6.2.2 抛物线插值及其 MATLAB 程序

定理 6.2 设函数 $f(x)$ 在 $[a,b]$ 上具有三阶连续导数，且满足条件

$$f(x_0)=y_0,\quad f(x_1)=y_1,\quad f(x_2)=y_2, \tag{6.12}$$

则

(1) 存在二次插值多项式

$$L_2(x)=y_0l_0+y_1l_1+y_2l_2, \tag{6.13}$$

其中

$$l_0=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)},\quad l_1=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)},\quad l_2=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}, \tag{6.14}$$

且满足

$$L_2(x_0)=y_0,\quad L_2(x_1)=y_1,\quad L_2(x_2)=y_2. \tag{6.15}$$

(2) 二次插值多项式(6.13)对于任意 $x\in(a,b)$，有插值余项

$$R_2(x)=f(x)-L_2(x)=\frac{f'''(\xi)}{3!}(x-x_2)(x-x_1)(x-x_0),\quad \xi\in(a,b). \tag{6.16}$$

证明 (1) 首先求作二次式 l_0，使满足

$$l_0(x_j)=\begin{cases}1,j=0,\\0,j\neq 0\end{cases}\quad (j=1,2). \tag{6.17}$$

在(6.17)式中，当 $j\neq 0\ (j=1,2)$ 时，$l_0(x_j)=0$，知 x_1,x_2 是 $l_0(x)$ 的零点，得

$$l_0=c(x-x_1)(x-x_2).$$

再利用 $l_0(x_0)=1$ 代入上式，得

$$l_0=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}.$$

类似地，可以构造出满足条件

$$l_1(x_j)=\begin{cases}1,j=1,\\0,j\neq1\end{cases}\quad (j=0,2)\text{ 和 } l_2(x_j)=\begin{cases}1,j=2,\\0,j\neq2\end{cases}\quad (j=0,1)$$

的函数分别为

$$l_1(x)=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)},l_2(x)=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}.$$

将已知数据 y_0,y_1,y_2 作为系数,得二次函数(6.13). 把 x_0,x_1,x_2 分别代入(6.13)式,得(6.15)式.

(2) 根据定理 6.1 可知,插值余项(6.17)成立. 证毕

这种二次插值(6.13)称为**抛物线插值函数**,也称为**三点二次插值函数**,(6.14)式称**抛物线插值基函数**,用 $L_2(x)$ 计算插值称**抛物线插值**.

例 6.2.3 求将区间$[0,\pi/2]$分成 n 等份($n=1,2$),用 $y=f(x)=\cos x$ 产生 $n+1$ 个节点,然后根据(6.9)和(6.13)式分别作线性插值函数 $P_1(x)$和抛物线插值函数 $P_2(x)$. 用它们分别计算 $\cos(\pi/6)$(取四位有效数字),并估计其误差.

解 下面分别用手工计算和 MATLAB 程序计算.

(1) **方法 1**(手工计算) 若 $n=1$,则$(x_0,y_0)=(0,1)$,$(x_1,y_1)=(\pi/2,0)$. 由(6.9)式,得

$$L_1(x)=y_0l_0+y_1l_1=1\cdot\frac{x-\pi/2}{0-\pi/2}+0\cdot\frac{x-0}{\pi/2-0}=1-2x/\pi,$$

$$\cos(\pi/6)\approx L_1(\pi/6)=0.666\ 7.$$

下面估计它们所产生的误差$|R_1(x)|$. 对于 $f(x)=\cos x$,可设$|f''(\xi)|=|-\cos\xi|\leqslant M_2=1$,根据(6.11),得

$$|R_1(\pi/6)|=\frac{1}{2!}|f''(\xi)||(\pi/6-\pi/2)(\pi/6-0)|\leqslant 0.274\ 2,\xi\in(0,\pi/2).$$

精确值 $\cos(\pi/6)=0.866\ 0$(4 位有效数字). $L_1(\pi/6)$的误差在$|R_1(x)|$范围内.

方法 2(MATLAB 程序计算) 输入程序

```
>> X=[0,pi/2]; Y =cos(X) ,
l01 =poly(X(2))/( X(1) - X(2)),l11 =poly(X(1))/( X(2) - X(1)),
l0 =poly2sym (l01),
l1 =poly2sym (l11),P =l01 * Y(1) +l11 * Y(2),L =poly2sym (P),x =pi/6;
Y =polyval(P,x)
```

运行后输出基函数 l_0 和 l_1 及其插值多项式的系数向量 $\boldsymbol{P}$、插值多项式和插值为

```
l0 =
    -5734161139222659/9007199254740992 * x +1
l1 =
    5734161139222659/9007199254740992 * x
```

```
P =
     -0.6366    1.0000
L =
     -5734161139222659/9007199254740992*x+1
Y =
    0.6667
```

输入程序

```
>> M=1;x=pi/6; R1=M*abs((x-X(1))*(x-X(2)))/2
```

运行后输出误差限为

```
R1 =
    0.2742.
```

(2) **方法1**(手工计算) 若 $n=2$,则 $(x_0,y_0)=(0,1)$,$(x_1,y_1)=(\pi/4, 0.7071)$,$(x_2,y_2)=(\pi/2,0)$,由(6.13),(6.14)式,得

$$L_2(x)=y_0l_0+y_1l_1+y_2l_2=8(x-\pi/4)(x-\pi/2)/\pi^2-16x(x-\pi/2)0.7071/\pi^2,$$

$$\cos(\pi/6)\approx L_2(\pi/6)=0.8508.$$

下面估计它所产生的误差 $|R_2(x)|$,对于 $f(x)=\cos x$,可设 $|f'''(\xi)|=|\sin\xi|\leqslant M_3=1$,根据(6.16)式,得

$$|R_2(x)|=\frac{|f'''(\xi)|}{3!}|(\pi/6-\pi/2)(\pi/6-\pi/4)(\pi/6-0)|\leqslant 0.0239,$$

所以 $L_2(\pi/6)$ 的误差在 $|R_2(x)|$ 范围内.

方法2(MATLAB 程序计算) 输入程序

```
>> X=0:pi/4:pi/2; Y=cos(X),
l01=conv (poly(X(2)),
poly(X(3)))/(( X(1) - X(2)) * ( X(1) - X(3))),
l11=conv (poly(X(1)),
poly(X(3)))/(( X(2) - X(1)) * ( X(2) - X(3))),
l21=conv (poly(X(1)),
poly(X(2)))/(( X(3) - X(1)) * ( X(3) - X(2))),
l0=poly2sym (l01),l1=poly2sym (l11),l2=poly2sym (l21),
P=l01* Y(1) + l11* Y(2)  + l21* Y(3), L=poly2sym (P),x=pi/6;
Y=polyval(P,x)
```

运行后输出基函数 l_0,l_1 和 l_2 及其插值多项式的系数向量 $\boldsymbol{P}$、插值多项式 L 和插值 Y 为

```
l0 =
      228155022448185/281474976710656*x^2 -2150310427208497/
1125899906842624*x+1
l1 =
```

```
        -228155022448185/140737488355328*x^2+5734161139222659/
2251799813685248*x
    l2 =
        228155022448185/281474976710656*x^2-5734161139222659/
9007199254740992*x
    P =
        -0.3357   -0.1092   1.0000
    L =
         -6048313895780875/18014398509481984*x^2-7870612110600739/
72057594037927936*x+1
    Y -
       0.8508
```

输入程序

```
>> M=1;x=pi/6; R2=M*abs((x-X(1))*(x-X(2)) *(x-X(3)))/6
```

运行后输出误差限为

```
    R2 -
       0.0239.
```

6.2.3 n 次拉格朗日插值及其 MATLAB 程序

我们可以根据定理 6.1 将抛物线插值的结论推广到 n 次插值多项式,有下面的定理.

定理 6.3 设 $f(x)$ 在 $[a,b]$ 上具有 $n+1$ 阶连续导数,对于 $n+1$ 个节点 (x_j, y_j), $j=0,1,\cdots,n$,其中 x_j 互不相同,满足

$$f(x_j)=y_j,\quad j=0,1,\cdots,n$$

则(1)存在 n 次插值多项式

$$L_n(x) = \sum_{i=0}^{n} y_i l_i(x), \tag{6.18}$$

其中 n 次多项式

$$l_i(x)=\frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}, i=0,1,\cdots,n, \tag{6.19}$$

满足

$$l_i(x_j)=\begin{cases}1, i=j,\\0, i\neq j,\end{cases}\quad i,\quad j=0,1,\cdots,n, \tag{6.20}$$

且

$$L_n(x_j)=y_j,\quad j=0,1,\cdots,n.$$

(2) n 次插值多项式(6.18)对于任意 $x\in(a,b)$,有插值余项

$$R_n(x)=f(x)-L_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_n)\cdots(x-x_1)(x-x_0),\xi\in(a,b) \tag{6.21}$$

(3) 如果存在一个正数 M_{n+1},使得 $|f^{(n+1)}(\xi)|\leqslant M_{n+1}$,则

$$|R_n(x)|\leqslant\frac{M_{n+1}}{(n+1)!}|(x-x_n)\cdots(x-x_1)(x-x_0)|. \tag{6.22}$$

我们可以用定理 6.2 证明的方法证明定理 6.3. 根据定理 6.1,得(6.18)式表示的 $L_n(x)$ 与(6.1)式相同. (6.19)式称**基函数**,(6.18)式和(6.19)式称**拉格朗日插值多项式**,用 $L_n(x)$ 计算插值称**拉格朗日多项式插值**.

当 $n=1$ 时,(6.18)式是满足条件 $L_1(x_0)=y_0,L_1(x_1)=y_1$ 的线性插值函数(6.9).

当 $n=2$ 时,(6.18)式是满足条件 $L_2(x_0)=y_0,L_2(x_1)=y_1,L_2(x_2)=y_2$ 的抛物线插值函数(6.13).

例 6.2.4 给出节点数据 $f(-2.00)=17.00$, $f(0.00)=1.00$, $f(1.00)=2.00$, $f(2.00)=17.00$,作三次拉格朗日插值多项式计算 $f(0.6)$,并估计其误差.

解 **方法 1**(手工计算) 取 $x_0=-2.00,y_0=17.00,x_1=0.00,y_1=1.00,x_2=1.00,y_2=2.00,x_3=2.00,y_3=17.00$,则基函数为

$$l_0(x)=\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}=-\frac{1}{24}x(x-1)(x-2),$$

$$l_1(x)=\frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}=\frac{1}{4}(x+2)(x-1)(x-2),$$

$$l_2(x)=\frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}=-\frac{1}{3}x(x+2)(x-2),$$

$$l_3(x)=\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}=\frac{1}{8}x(x-1)(x+2).$$

根据(6.18)式,得

$$\begin{aligned}P_3(x)&=\sum_{i=0}^{3}y_il_i(x)\\&=-\frac{17}{24}x(x-1)(x-2)+\frac{1}{4}(x+2)(x-1)(x-2)-\\&\quad\frac{2}{3}x(x+2)(x-2)+\frac{17}{8}x(x-1)(x+2),\end{aligned}$$

$f(0.6)\approx P_3(0.6)=0.256\approx0.26.$

根据(6.21)式知,对于任意 $x\in(-2.00,2.00)$,有插值余项

$$R_3(x)=f(x)-P_3(x)=\frac{f^{(4)}(\xi)}{4!}x(x-1)(x-2)(x+2),\xi\in(-2.00,2.00),$$

$$R_3(0.6)=\frac{f^{(4)}(\xi)}{4!}\cdot 0.6\cdot(0.6-1)(0.6-2)(0.6+2)$$

$$\approx 0.036f^{(4)}(\xi)\approx 0.04f^{(4)}(\xi).$$

如果存在一个正数 M_4，使得 $|f^{(4)}(\xi)|\leqslant M_4$，由(6.22)式，得

$$|R_3(0.6)|\leqslant 0.04M_4,\quad \xi\in(-2.00,2.00).$$

方法 2(MATLAB 程序计算)　输入程序

```
>> X=[-2,0,1,2]; Y =[17,1,2,17];
p1=poly(X(1)); p2=poly(X(2)); p3=poly(X(3)); p4=poly(X(4));
l01=conv(conv(p2,p3),p4)/((X(1)-X(2))*(X(1)-X(3))*
(X(1)-X(4))),
l11=conv(conv(p1,p3),p4)/((X(2)-X(1))*(X(2)-X(3))*
(X(2)-X(4))),
l21-conv(conv(p1,p2),p4)/((X(3)-X(1))*(X(3)-X(2))*
(X(3)-X(4))),
l31-conv(conv(p1,p2),p3)/((X(4)  X(1))*(X(4)  X(2))*
(X(4)-X(3))),
l0=poly2sym(l01),l1=poly2sym(l11),l2=poly2sym(l21),l3=
poly2sym(l31),
P=l01*Y(1)+l11*Y(2) +l21*Y(3) +l31*Y(4),
```

运行后输出基函数 l_0,l_1,l_2 和 l_3 及其插值多项式的系数向量 $\boldsymbol{P}$（略）为

```
l0 =
     -1/24*x^3+1/8*x^2-1/12*x,l1=1/4*x^3-1/4*x^2-x+1
l2 =
     -1/3*x^3+4/3*x,l3=1/8*x^3+1/8*x^2-1/4*x
```

输入程序

```
>> L=poly2sym(P),x=0.6; Y=polyval(P,x)
```

运行后输出插值多项式 L 和插值 Y 为

```
L =                                    Y =
    x^3+4*x^2-4*x+1                        0.2560.
```

输入程序

```
>> syms M; x=0.6;
   R3=M*abs((x-X(1))*(x-X(2)) *(x-X(3)) *(x-X(4)))/24
```

运行后输出误差限为

```
R3 =
    91/2500*M
```

即

$$R_3 = 0.0364 f^{(4)}(\xi), \quad \xi \in (-2.00, 2.00).$$

6.2.4 事后估计方法

定理6.1给出了插值的误差通过插值多项式 $P_n(x)$ 与产生节点 (x_j, y_j) 的 $f(x)$ 之差来估计 $R_n(x)$ 的公式(6.5)和(6.6). 如果已知 $f(x)$ 的解析表达式且具有 $n+1$ 阶导数,或者不知道 $f(x)$ 的解析表达式,但是存在一个正数 M_{n+1},使得 $|f^{(n+1)}(\xi)| \leqslant M_{n+1}$,则可以根据(6.5)或(6.6)式估计误差 $R_n(x)$ 的大小. 但是,在实际计算中,并不知道 $f(x)$ 的解析表达式,难以得到 $f^{(n+1)}(\xi)$ 的形式或较精确的界 M_{n+1}. 因此,很难得到 $|R_n(x)|$ 的大小或者 $|R_n(x)|$ 的界. 但是可以看出,n 增加,$|R_n(x)|$ 减少;$f(x)$ 越光滑,M_{n+1} 越小,$|R_n(x)|$ 越小;x 越接近 x_j,$|R_n(x)|$ 越小.

在实际计算中,可对误差运用下面的事后估计方法. 设 $f(x)$ 在 $[a,b]$ 上具有 $n+1$ 阶连续导数,而且 $f^{(n+1)}(x)$ 变化不大,对于 $n+2$ 个节点 (x_j, y_j), $j=0,1,\cdots,n,n+1$,其中 x_j 互不相同,满足 $f(x_j)=y_j$, $j=0,1,\cdots,n,n+1$. 任选其中的 $n+1$ 个节点,不妨取 (x_j,y_j), $j=0,1,\cdots,n$,构造一个 n 次插值多项式 $L_n(x) = \sum_{i=0}^{n} y_i l_i(x)$. 在 $n+2$ 个节点 (x_j,y_j), $j=0,1,\cdots,n,n+1$ 中选择其中的 $n+1$ 个节点,不妨取 (x_j,y_j), $j=1,2,\cdots,n,n+1$,构造另一个 n 次插值多项式 $\tilde{L}_n(x) = \sum_{i=1}^{n+1} y_i \tilde{l}_i(x)$. 根据定理6.3可得

$$R_n(x) = f(x) - L_n(x) = \frac{f^{(n+1)}(\xi_1)}{(n+1)!}(x-x_n)\cdots(x-x_1)(x-x_0), \xi_1 \in (a,b),$$

$$\tilde{R}_n(x) = f(x) - \tilde{L}_n(x) = \frac{f^{(n+1)}(\xi_2)}{(n+1)!}(x-x_{n+1})\cdots(x-x_2)(x-x_1), \xi_2 \in (a,b),$$

因为 $f^{(n+1)}(x)$ 在 $[a,b]$ 上连续而且变化不大,所以 $f^{(n+1)}(\xi_1) \approx f^{(n+1)}(\xi_2)$,则

$$\frac{f(x) - L_n(x)}{f(x) - \tilde{L}_n(x)} \approx \frac{x - x_0}{x - x_{n+1}},$$

从而可得到

$$f(x) \approx \frac{x - x_{n+1}}{x_0 - x_{n+1}} L_n(x) + \frac{x - x_0}{x_{n+1} - x_0} \tilde{L}_n(x), \tag{6.23}$$

$$f(x) - L_n(x) \approx \frac{x - x_0}{x_{n+1} - x_0} [\tilde{L}_n(x) - L_n(x)]. \tag{6.24}$$

按估计式(6.23),插值结果 $L_n(x)$ 的误差 $f(x) - L_n(x)$ 可以通过两个结果的偏差 $L_n(x) - \tilde{L}_n(x)$ 来估计. 这种直接用计算结果估计误差的方法称作**事后估计法**.

6.2.5　拉格朗日多项式和基函数的 MATLAB 程序

按照(6.18)、(6.19)式，现提供一个求拉格朗日插值多项式和基函数的主程序，保存名为 lagran1.m 的 M 文件.

求拉格朗日插值多项式和基函数的 MATLAB 主程序

输入的量：$n+1$ 个节点$(x_i,y_i)(i=1,2,\cdots,n+1)$横坐标向量 $\boldsymbol{X}$，纵坐标向量 $\boldsymbol{Y}$；

输出的量：n 次拉格朗日插值多项式 L 及其系数向量 $\boldsymbol{C}$，基函数 l 及其系数矩阵 $\boldsymbol{L}_1$.

```
function [C, L,L1,l] = lagran1(X,Y)
m = length(X); L = ones(m,m);
for k = 1:m
    V = 1;
    for i = 1:m
    if k ~ = i
          V = conv(V,poly(X(i)))/(X(k) - X(i));
    end
end
L1(k,:) = V; l(k,:) - poly2sym (V)
end
C = Y * L1;L = Y * l
```

例 6.2.5　给出节点数据 $f(-2.15)=17.03$，$f(-1.00)=7.24$，$f(0.01)=1.05$，$f(1.02)=2.03$，$f(2.03)=17.06$，$f(3.25)=23.05$，作五次拉格朗日插值多项式和基函数，并写出估计其误差的公式.

解　(1) 保存名为 lagran1.m 的 M 文件.

(2) 在 MATLAB 工作窗口输入程序

```
>> X = [ -2.15  -1.00 0.01 1.02 2.03 3.25];
   Y = [17.03 7.24 1.05 2.03 17.06 23.05];
   [C, L ,L1,l] = lagran1(X,Y)
```

(3) 运行后输出五次拉格朗日插值多项式 L 及其系数向量 $\boldsymbol{C}$，基函数 l 及其系数矩阵 $\boldsymbol{L}_1$ 如下

```
C =
     -0.2169   0.0648   2.1076   3.3960   -4.5745   1.0954
L -
     1.0954 -4.5745 * x + 3.3960 * x^2 + 2.1076 * x^3 + 0.0648 * x^4 -
0.2169 * x^5
```

```
L1 =
   -0.0056    0.0299   -0.0323   -0.0292    0.0382   -0.0004
    0.0331   -0.1377   -0.0503    0.6305   -0.4852    0.0048
   -0.0693    0.2184    0.3961   -1.2116   -0.3166    1.0033
    0.0687   -0.1469   -0.5398    0.6528    0.9673   -0.0097
   -0.0317    0.0358    0.2530   -0.0426   -0.2257    0.0023
    0.0049    0.0004   -0.0266    0.0001    0.0220   -0.0002
l =
   [ -0.0056*x^5+0.0299*x^4-0.0323*x^3-0.0292*x^2+
0.0382*x-0.0004]
   [ 0.0331*x^5-0.1377*x^4-0.0503*x^3+0.6305*x^2-0.4852
*x+0.0048]
   [ -0.0693*x^5+0.2184*x^4+0.3961*x^3-1.2116*x^2-
0.3166*x+1.0033]
   [ 0.0687*x^5-0.1469*x^4-0.5398*x^3+0.6528*x^2+0.9673
*x-0.0097]
   [ -0.0317*x^5+0.0358*x^4+0.2530*x^3-0.0426*x^2-
0.2257*x+0.0023]
   [ 0.0049*x^5+0.0004 *x^4-0.0266*x^3+0.0001*x^2+0.0220
*x-0.0002]
```

估计其误差的公式为

$$R_5(x)=\frac{f^{(6)}(\xi)}{6!}(x+2.15)(x+1.00)(x-0.01)(x-1.02)(x-2.03)(x-3.25),$$
$$\xi\in(-2.15,3.25).$$

6.2.6 拉格朗日插值及其误差估计的 MATLAB 程序

设$f(x)$在$[a,b]$上具有 $n+1$ 阶连续导数，对于 $n+1$ 个节点(x_j,y_j)，$j=0,1\cdots,n$，其中 x_j 互不相同，满足$f(x_j)=y_j$，$j=0,1,\cdots,n$. 则按照(6.18)，(6.19)和(6.22)式，改写求拉格朗日插值及其误差估计的主程序，保存名为 lagranzi.m 的 M 文件.

拉格朗日插值及其误差估计的 MATLAB 主程序

输入的量: X 是 $n+1$ 个节点$(x_i,y_i)(i=1,2,\cdots,n+1)$横坐标向量，$Y$ 是纵坐标向量，x 是以向量形式输入的 m 个插值点，M 在$[a,b]$上满足 $|f^{(n+1)}(x)|\leqslant M$.

输出的量: y 为 m 个插值构成的向量，R 是误差限.

```
function [y,R] = lagranzi(X,Y,x,M)
```

```
n = length(X); m = length(x);
for i = 1:m
    z = x(i);s = 0.0;
    for k = 1:n
          p = 1.0; q1 = 1.0; c1 = 1.0;
        for j = 1:n
          if j ~ = k
          p = p*(z - X(j))/(X(k) - X(j));
          end
      q1 = abs(q1 * (z - X(j)));c1 = c1 * j;
      end
      s = p * Y(k) + s;
    end
    y(i) = s;
end
R = M * q1 / c1;
```

例 6.2.6 已知 $\sin 30° = 0.5$, $\sin 45° = 0.707\,1$, $\sin 60° = 0.866\,0$,用拉格朗日插值及其误差估计的 MATLAB 主程序求 $\sin 40°$的近似值,并估计其误差.

解 (1) 取 $f(x) = \sin x$,则 $f'''(x) = -\cos x$, $|f'''(x)| = |-\cos x| \leqslant 1 = M$, $30° = \pi/6$, $45° = \pi/4$, $60° = \pi/3$,求 $40° = 2\pi/9$.

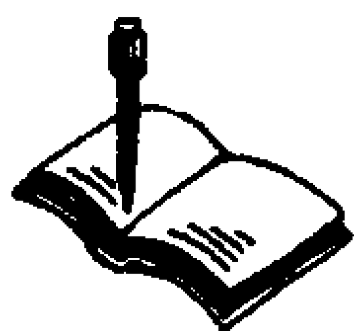
(2) 保存名为 lagranzi.m 的 M 文件.

(3) 在 MATLAB 工作窗口输入程序

```
>> x = 2 * pi / 9; M = 1; X = [pi / 6 ,pi / 4, pi / 3];
   Y = [0.5,0.7071,0.8660]; [y,R] = lagranzi(X,Y,x,M)
```

(4) 运行后输出插值 y 及其误差限 R 为

```
y =                    R =
   0.6434                 8.8610e - 004.
```



习 题 6.2

1. 已知函数 $f(x)$在$[1,7]$上具有二阶连续导数,$|f''(x)| \leqslant 5$,且满足条件 $f(1) = 1$, $f(7) = 12$. 求线性插值多项式和函数值 $f(3.5)$,并估计其误差.

2. 求函数 $f(x) = e^{-3x}$在$[0,4]$上的线性插值多项式,并估计其误差.

3. 求将区间$[\pi/6, \pi/2]$分成 n 等份($n = 1,2$),用 $y = f(x) = \sin x$ 产生 $n + 1$ 个节点,然后根据(6.9)和(6.13)式分别作线性插值函数 $P_1(x)$和抛物线插值函数 $P_2(x)$. 用它们分别计算 $\sin(\pi/5)$ (取四位有效数字),并估计其误差.

4. 给出节点数据 $f(-3.00)=27.00$, $f(0.00)=1.00$, $f(1.00)=2.00$, $f(2.00)=17.00$,作二次拉格朗日插值多项式计算 $f(1.4)$,并估计其误差.

5. 给出节点数据 $f(-3.15)=37.03$, $f(-1.00)=7.24$, $f(0.01)=1.05$, $f(1.02)=2.03$, $f(2.03)=17.06$, $f(3.25)=23.05$,作五次拉格朗日插值多项式和基函数,并写出估计其误差的公式.

6. 已知 $\sin 30°=0.5$, $\sin 45°=0.7071$, $\sin 90°=1$,求 $\sin 40°$的近似值,并估计其误差.

6.3 牛顿(Newton)插值及其 MATLAB 程序

拉格朗日插值的优点是格式整齐和规范,有误差估计公式,它的缺点是没有承袭性,当需要增加节点时,必须重新计算插值的基函数 $l_i(x)$. 本节给出具有承袭性的牛顿插值法及其 MATLAB 程序. 为此,首先介绍与牛顿插值有关的差商等概念、性质及其计算.

6.3.1 差商及其计算

差商也称均差. 它的作用与拉格朗日插值的基函数一样,差商是构造牛顿插值的基础.

定义 6.1 设函数 $f(x)$ 在 $[a,b]$ 上有定义, $x_i, x_j \in [a,b]$ 且 $x_i \neq x_j$,则

$$f[x_i,x_j]=\frac{f(x_i)-f(x_j)}{x_i-x_j} \tag{6.25}$$

称为函数 $f(x)$ 在点 x_i, x_j 的**一阶差商**.

设 $f[x_i,x_j]$ 和 $f[x_j,x_k]$ 为函数 $f(x)$ 在点 x_i, x_j 和 x_j, x_k 的一阶差商,且 $x_i \neq x_k$,则

$$f[x_i,x_j,x_k]=\frac{f[x_i,x_j]-f[x_j,x_k]}{x_i-x_k} \tag{6.26}$$

称为函数 $f(x)$ 在点 x_i, x_j, x_k 的**二阶差商**.

一般的,设 $f[x_0,x_1,\cdots,x_{k-1}]$ 和 $f[x_1,x_2,\cdots,x_k]$ 为函数 $f(x)$ 在点 $x_0,x_1,\cdots,x_{k-1}$ 和 $x_1,x_2,\cdots,x_k$ 的 $k-1$ 阶差商,且 $x_0 \neq x_k$,则

$$f[x_0,x_1,\cdots,x_k]=\frac{f[x_0,x_1,\cdots,x_{k-1}]-f[x_1,x_2,\cdots,x_k]}{x_0-x_k} \tag{6.27}$$

称为函数 $f(x)$ 在点 $x_0,x_1,x_2,\cdots,x_k$ 的 **k 阶差商**.

函数 $f(x)$ 在节点处函数值 $f(x_0)$, $f(x_1)$, …称**零阶插商**.

差商具有如下的基本性质:

性质 1 n 次多项式 $P_n(x)$ 的 k 阶差商

$$P_n[x_0,x_1,x_2,\cdots,x_k]=\begin{cases}0, & \text{当 } k>n \text{ 时},\\ n-k \text{ 次多项式}, & \text{当 } k\leqslant n \text{ 时}.\end{cases}$$

例如,如果$f(x)$是n次多项式,则$f(x)$的$n+1$阶差商$f[x,x_0,x_1,\cdots,x_k]\equiv 0$.

性质 2 函数$f(x)$的n阶差商$f[x_0,x_1,\cdots,x_n]$是零阶插商(函数值)$f(x_0)$,$f(x_1)$,…,$f(x_n)$的线性组合.即

$$f[x_0,x_1,\cdots,x_n]=\sum_{k=0}^{n}\frac{f(x_k)}{\tilde{\omega}'(x_k)},\text{其中}\ \tilde{\omega}'(x_k)=\prod_{\substack{i=0\\i\neq k}}^{n}(x_k-x_i).$$

例如,$f[x_0,x_1]=\dfrac{f(x_1)-f(x_0)}{x_1-x_0}=\dfrac{f(x_1)}{x_1-x_0}+\dfrac{f(x_0)}{x_0-x_1}=\sum\limits_{k=0}^{1}\dfrac{f(x_k)}{\tilde{\omega}'(x_k)}$.

性质 3 差商关于节点具有对称性,即与节点的排列顺序无关.

例如,$f[x_0,x_1]=f[x_1,x_0]$,

$$\begin{aligned}f[x_0,x_1,x_2]&=f[x_1,x_2,x_0]=f[x_0,x_2,x_1]=f[x_1,x_0,x_2]\\&=f[x_2,x_1,x_0]=f[x_2,x_0,x_1].\end{aligned}$$

性质 3 可以由性质 2 推出.

性质 4 如果$f[x,x_0,x_1,\cdots,x_k]$是x的m次多项式,则$f[x,x_0,x_1,\cdots,x_k,x_{k+1}]$是$x$的$m-1$次多项式($m\geqslant 1$).

证明 由差商的定义

$$f[x,x_0,x_1,\cdots,x_k,x_{k+1}]=\frac{f[x_0,x_1,\cdots,x_k,x_{k+1}]-f[x,x_0,x_1,\cdots,x_k]}{x_{k+1}-x},$$

右端分子为m次多项式,且当$x=x_{k+1}$时,分子为零,故分子含因子$x_{k+1}-x$,它与分母相消后,右端为$m-1$次多项式.证毕

按照差商的定义,用两个$k-1$阶差商的值计算k阶差商,通常用插商表的形式计算和存放(见表 6-1).

表 6-1 插 商 表

x_k	$f(x_k)$	一阶差商	二阶差商	三阶差商	…	n阶差商
x_0	$f(x_0)$					
x_1	$f(x_1)$	$f[x_0,x_1]$				
x_2	$f(x_2)$	$f[x_1,x_2]$	$f[x_0,x_1,x_2]$			
x_3	$f(x_3)$	$f[x_2,x_3]$	$f[x_1,x_2,x_3]$	$f[x_0,x_1,x_2,x_3]$		
⋮	⋮	⋮	⋮	⋮	⋱	
x_n	$f(x_n)$	$f[x_{n-1},x_n]$	$f[x_{n-2},x_{n-1},x_n]$	$f[x_{n-3},x_{n-2},x_{n-1},x_n]$	…	$f[x_0,x_1,\cdots,x_n]$

例 6.3.1 计算$(-2,17)$,$(0,1)$,$(1,2)$,$(2,19)$的一至三阶差商.

解 $f[-2,0]=\dfrac{f(0)-f(-2)}{0-(-2)}=\dfrac{1-17}{0-(-2)}=-8$,

$$f[1,0]=\frac{f(0)-f(1)}{0-1}=\frac{1-2}{0-1}=1,$$

$$f[1,2]=\frac{f(2)-f(1)}{2-1}=\frac{19-2}{2-1}=17,$$

$$f[-2,0,1]=\frac{f[0,1]-f[-2,0]}{1-(-2)}=\frac{1+8}{1-(-2)}=3,$$

$$f[0,1,2]=\frac{f[1,2]-f[0,1]}{2-0}=\frac{17-1}{2-0}=8,$$

$$f[-2,0,1,2]=\frac{f[0,1,2]-f[-2,0,1]}{2-(-2)}=\frac{8-3}{4}=\frac{5}{4}.$$

将上述的计算结果列成差商表 6－2.

表 6－2 例 6.3.1 的差商表

x_n	$f(x_n)$	$f[x_{n-1},x_n]$	$f[x_{n-2},x_{n-1},x_n]$	$f[x_{n-3},x_{n-2},x_{n-1},x_n]$
−2	17			
0	1	−8		
1	2	1	3	
2	19	17	8	5/4

6.3.2 牛顿插值

设函数 $f(x)$ 在 $[a,b]$ 上有定义，对于 $n+1$ 个节点 (x_j,y_j)，$j=0,1,\cdots,n$，其中 $x_j\in[a,b]$ 互不相同，满足 $f(x_j)=y_j$，$j=0,1,\cdots,n$. 对任意 $x\in[a,b]$，由一阶差商的定义 $f[x,x_0]=\dfrac{f(x)-f(x_0)}{x-x_0}$，得

$$f(x)=f(x_0)+f[x,x_0](x-x_0).$$

类似地，由二阶差商至 n 阶差商的定义得到下列方程

$$f[x,x_0]=f[x_0,x_1]+f[x,x_0,x_1](x-x_1),$$

$$f[x,x_0,x_1]=f[x_0,x_1,x_2]+f[x,x_0,x_1,x_2](x-x_2),$$

…………

$$f[x,x_0,x_1,\cdots,x_{n-1}]=f[x_0,x_1,x_2,\cdots,x_n]+f[x,x_0,x_1,\cdots,x_n](x-x_n).$$

反复用后一个式子代入前面的式子，整理后得到

$$\begin{aligned}f(x)=&f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)+\cdots+\\&f[x_0,x_1,x_2,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})+\\&f[x,x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})(x-x_n).\end{aligned}\tag{6.28}$$

(6.28)式称**带余项的牛顿插值公式**.

令

$$P_n(x)=f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)+\cdots+ f[x_0,x_1,x_2,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1}), \tag{6.29}$$

$$R_n(x)=f[x,x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})(x-x_n), \tag{6.30}$$

则带余项的牛顿插值公式(6.28)可以表示为

$$f(x)=P_n(x)+R_n(x),$$

其中 $P_n(x)$ 为次数不高于 n 次的 x 的多项式,可以验证 $P_n(x_i)=f(x_i)\ (i=0,1,\cdots,n)$. $P_n(x)$ 称为过 $n+1$ 个节点的 $f(x)$ 的 n 阶**牛顿插值多项式**,$R_n(x)$ 称为**牛顿插值多项式的余项**.

由(6.29)式可见,牛顿插值多项式具有承袭性,增加一个节点时只增加一项. 根据定理 6.1 知,$n+1$ 个节点可以确定唯一的 n 次插值多项式,即此时拉格朗日插值多项式与牛顿插值多项式是相等的,只是形式上不同而已,因此它们的余项也相等. 综上所述可得到下面的定理.

定理 6.4 设函数 $f(x)$ 在 $[a,b]$ 上具有 $n+1$ 阶连续的导数,对于 $[a,b]$ 上的 $n+1$ 个横坐标不同的节点 (x_j,y_j), $j=0,1,\cdots,n$, 满足 $f(x_j)=y_j, j=0,1,\cdots,n$. 则存在次数不高于 n 次的唯一的 x 的牛顿插值多项式(6.29),使得 $P_n(x_j)=f(x_j)\ (j=1,2,\cdots,n+1)$,且

$$f(x)=P_n(x)+R_n(x),$$

其中 $R_n(x)$ 与拉格朗日插值余项相同,即

$$R_n(x)=f[x,x_0,x_1,\cdots,x_n]\prod_{k=0}^{n}(x-x_k)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{k=0}^{n}(x-x_k), \xi\in(a,b). \tag{6.31}$$

例 6.3.2 求过节点 $(-2,17)$, $(0,1)$, $(1,2)$, $(2,19)$ 的牛顿插值多项式 $P_n(x)$,并计算 $P_2(0.9)$ 和 $P_3(0.9)$.

解 (1) 首先计算 $P_2(0.9)$.

把 $x_0=-2, x_1=0, x_2=1$ 和例 6.3.1 差商表 6-2 中主对角线上的值 $f(x_0)=17$, $f[x_0,x_1]=-8$, $f[x_0,x_1,x_2]=3$ 代入二阶牛顿插值多项式

$$P_2(x)=f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1),$$

得

$$P_2(x)=17-8(x+2)+3(x+2)x,$$

故

$$P_2(0.9)=17-8\cdot(0.9+2)+3\cdot(0.9+2)\cdot 0.9=1.63.$$

(2) 再计算 $P_3(0.9)$. 根据(6.29)式,得三阶牛顿插值多项式

$$P_3(x)=P_2(x)+f[x_0,x_1,x_2,x_3](x-x_0)(x-x_1)(x-x_2).$$

将 $x_0=-2,x_1=0,x_2=1,x_3=2$ 和 $f[x_0,x_1,x_2,x_3]=\frac{5}{4}$ 代入上式,得

$$P_3(x)=17-8(x+2)+3x(x+2)+\frac{5}{4}(x+2)(x-0)(x-1).$$

因此,$P_3(0.9)=P_2(0.9)+\frac{5}{4}(0.9+2)(0.9-0)(0.9-1)\approx 1.30.$

6.3.3 牛顿插值多项式、差商和误差公式的 MATLAB 程序

按照差商的定义和(6.29)式,现提供求牛顿插值多项式和差商的 MATLAB 主程序,保存名为 newpoly.m 的 M 文件.

求牛顿插值多项式和差商的 MATLAB 主程序

输入的量:$n+1$ 个节点 (x_i,y_i) $(i=1,2,\cdots,n+1)$ 横坐标向量 $\boldsymbol{X}$,纵坐标向量 $\boldsymbol{Y}$.

输出的量:n 阶牛顿插值多项式 L 及其系数向量 $\boldsymbol{C}$,差商的矩阵 $\boldsymbol{A}$,插值余项公式 $wcgs$ 及其 $\frac{R_n(x)}{f^{(n+1)}(\xi)}$(其中 $\xi\in(\min\limits_{1\le i\le n+1}(x_i),\max\limits_{1\le i\le n+1}(x_i))$)的系数向量 $\boldsymbol{C}_w$.

```
function [A,C,L,wcgs,Cw] =newpoly (X,Y)
n =length(X); A =zeros(n,n); A(:,1) =Y';
s =0.0; p =1.0; q =1.0; c1 =1.0;
    for j =2:n
      for i =j:n
        A(i,j) =(A(i,j -1) - A(i -1,j -1))/(X(i) -X(i -j +1));
        end
      b =poly(X(j -1));q1 =conv(q,b); c1 =c1 * j; q =q1;
    end
  C =A(n,n); b =poly(X(n)); q1 =conv(q1,b);
for k =(n -1): -1:1
C =conv(C,poly(X(k))); d =length(C); C(d) =C(d) +A(k,k);
end
L(k,:) =poly2sym(C); Q =poly2sym(q1);
syms M
wcgs =M * Q/c1; Cw =q1/c1;
```

例 6.3.3 给出节点数据 $f(-2.15)=17.03$, $f(-1.00)=7.24$, $f(0.01)=1.05$, $f(1.02)=2.03$, $f(2.03)=17.06$, $f(3.25)=23.05$,作五阶牛顿插值多项式和差商,并写出其估计误差的公式.

解 (1) 保存名为 newpoly.m 的 M 文件.

(2) 输入 MATLAB 程序

```
>> X = [ -2.15   1.00 0.01 1.02 2.03 3.25];
Y = [17.03 7.24 1.05 2.03 17.06 23.05];
[A,C,L,wcgs,Cw] = newpoly (X,Y)
```

运行后输出差商矩阵 $\boldsymbol{A}$,五阶牛顿插值多项式 L 及其系数向量 $\boldsymbol{C}$,插值余项公式 *wcgs* 及其向量 $\boldsymbol{C}_w$ 如下

```
A =
   17.0300         0         0         0         0         0
    7.2400   -8.5130         0         0         0         0
    1.0500   -6.1287    1.1039         0         0         0
    2.0300    0.9703    3.5144    0.7604         0         0
   17.0600   14.8812    6.8866    1.1129    0.0843         0
   23.0500    4.9098   -4.4715   -3.5056   -1.0867   -0.2169
C =
   -0.2169    0.0648    2.1076    3.3960   -4.5745    1.0954
L =
 -7813219284746629/36028797018963968 * x^5 + 583849564517807/9007199254740992 * x ^ 4 + 593245028711263/ 281474976710656 * x ^ 3 + 3823593773002357/ 1125899906842624 * x ^ 2 - 321902673270315/70368744177664 * x + 308328649211299/281474976710656
wcgs =
1/720 * M * (x^6 - 79/25 * x^5 - 14201/2500 * x^4 + 4934097026900981/281474976710656 * x ^ 3 + 154500712237335/ 35184372088832 * x ^ 2 - 8170642380559269/562949953421312 * x + 5212760744134241/36028797018963968)
Cw =
    0.0014   -0.0044   -0.0079   0.0243   0.0061   -0.0202   0.0002
```

即

```
L = 1.0954 - 4.5745 * x + 3.3960 * x^2 + 2.1076 * x^3 + 0.0648 * x^4 - 0.2169 * x^5.
```

估计其误差的公式为

$$R_5(x) = \frac{f^{(6)}(\xi)}{6!}(x+2.15)(x+1.00)(x-0.01)(x-1.02)(x-2.03)(x-3.25),$$

$$\xi \in (-2.15, 3.25).$$

例 6.3.4 求函数 $f(x) = -7e^{-x/5}$ 在[2,6]上的五阶牛顿插值多项式,求其误差公式和误差限公式. 用它们计算 $f(3.156)$,并估计其误差.

解 (1) 输入 MATLAB 程序

```
>> X=2:4/5:6; Y=-7*exp(-X/5);
[A,C,L,wcgs,Cw] = newpoly (X,Y), x1=2:0.001:6; M=max(-7*exp(-x1/5)/(5^6)),
```

运行后输出差商矩阵 A，五阶牛顿插值多项式 L 及其系数向量 C，插值余项公式 *wcgs* 及其向量 C_w 如下

```
A =
   -4.6922        0         0         0         0         0
   -3.9985   0.8672         0         0         0         0
   -3.4073   0.7390   -0.0801         0         0         0
   -2.9035   0.6297   -0.0683    0.0049         0         0
   -2.4742   0.5366   -0.0582    0.0042   -0.0002         0
   -2.1084   0.4573   -0.0496    0.0036   -0.0002    0.0000
C =
   0.0000   -0.0004   0.0089   -0.1389   1.3985   -6.9991
L =
   9721799720875/1152921504606846976*x^5-3503994098647815/9223372036854775808*x^4+160742008798419/18014398509481984*x^3-1251152213853501/9007199254740992*x^2+6298131904328647/4503599627370496*x-3940156929554013/562949953421312
wcgs =
   1/720*M*(x^6-24*x^5+1172/5*x^4-5952/5*x^3+7276634802928539/2199023255552*x^2-5237461186650519/1099511627776*x+6085939356447121/2199023255552)
Cw =
   0.0014   -0.0333   0.3256   -1.6533   4.5959   -6.6159   3.8438
M =
   -1.3494e-004
```

(2) 输入 MATLAB 程序

```
>> syms x
wcgs1=1/720*M*1/720*M*(x^6-24*x^5+1172/5*x^4-5952/5*x^3+7276634802928539/2199023255552*x^2-5237461186650519/1099511627776*x+6085939356447121/2199023255552),
```

运行后输出误差限公式 *wcgs1* 如下

```
wcgs1 =
   5565367633581443/158456325028528675187087900672*x^6-16696102900744329/19807040628566084398385987584*x^5+1630652716639362799/198070406285660843983859875840*x^4-517579189923074199/12379400392853802748991242240*x^3+40497147813610772932297365501777/
```

```
34844914372704098658649559801013064853094 4 * x ^ 2 - 29148396970323855270001164718917/
17422457186352049329324779900506532426547 2 * x + 33870489914310283926665276375603/
34844914372704098658649559801013064853094 4
```

(3) 输入 MATLAB 程序

```
>> x = 3.156;
y = 9721799720875/1152921504606846976 * x^5 - 3503994098647815/
9223372036854775808 * x^4 + 160742008798419/18014398509481984 * x^3 -
1251152213853501/ 9007199254740992 * x ^ 2 + 6298131904328647/
4503599627370496 * x - 3940156929554013/562949953421312
wcgs2 = 1/720 * M * (x^6 - 24 * x^5 + 1172/5 * x^4 - 5952/5 * x^3 +
7276634802928539/2199023255552 * x^2 - 5237461186650519/1099511627776
* x + 6085939356447121/2199023255552)
```

运行后输出 $f(3.156)$ 的近似值 y 及其误差限 *wcgs2* 如下

```
y =
    -3.7237
wcgs2 =
    -2.4764e-007
```

6.3.4 牛顿插值及其误差估计的 MATLAB 程序

设 $f(x)$ 在 $[a,b]$ 上具有 $n+1$ 阶连续导数，对于在 $[a,b]$ 上的 $n+1$ 个节点 (x_j, y_j)，$j=0,1,\cdots,n$，其中 x_j 互不相同，满足 $f(x_j)=y_j, j=0,1,\cdots,n$. 在 6.3.3 目给出了求牛顿插值多项式和差商的 MATLAB 主程序；在这里给出另一种根据 (6.28)，(6.29) 和 (6.31) 式编写的求牛顿插值及其误差估计的主程序，保存名为 newcz.m 的 M 文件. 后者的 MATLAB 主程序如下：

> **牛顿插值及其误差估计的 MATLAB 主程序**
>
> **输入的量**：$\boldsymbol{X}$ 是 $n+1$ 个节点 (x_i, y_i) $(i=1,2,\cdots,n+1)$ 横坐标向量，$\boldsymbol{Y}$ 是纵坐标向量，$\boldsymbol{x}$ 是以向量形式输入的 m 个插值点，$\boldsymbol{M}$ 在 $[a,b]$ 上满足 $|f^{(n+1)}(x)| \leqslant M$.
>
> **输出的量**：向量 $\boldsymbol{y}$ 是向量 $\boldsymbol{x}$ 处的插值及其误差限 R.

```
function [y,R] = newcz(X,Y,x,M)
n = length(X); m = length(x);
for t = 1:m
    z = x(t); A = zeros(n,n); A(:,1) = Y';
    s = 0.0; p = 1.0; q1 = 1.0; c1 = 1.0;
        for j = 2:n
        for i = j:n
```

```
            A(i,j) = (A(i,j -1) - A(i -1,j -1))/(X(i) -X(i -j +1));
          end
        q1 = abs(q1 * (z - X(j -1)));c1 = c1 * j;
      end
      C = A(n,n);q1 - abs(q1 * (z - X(n)));
  for k = (n -1): -1:1
  C = conv(C,poly(X(k)));d - length(C); C(d) = C(d) + A(k,k);
  end
    y(k) = polyval(C, z);
  end
  R = M * q1 / c1;
```

例 6.3.5 已知 $\sin 30° = 0.5$, $\sin 45° = 0.707\ 1$, $\sin 60° = 0.866\ 0$,用牛顿插值法求 $\sin 40°$ 的近似值,估计其误差,并与例 6.2.6 的计算结果比较.

解 取 $f(x) = \sin x$,则 $f'''(x) = -\cos x$, $|f'''(x)| = |-\cos x| \leqslant 1 = M$, $30° = \pi/6$, $45° = \pi/4$, $60° = \pi/3$,求 $\sin 40° = \sin 2\pi/9$

方法 1(牛顿插值及其误差估计的 MATLAB 主程序) 输入 MATLAB 程序

```
>> x = 2 * pi / 9;M = 1; X = [pi/6 ,pi/4, pi/3];
Y = [0.5,0.7071,0.8660]; [y,R] = newcz(X,Y,x,M)
```

运行后输出插值 y 及其误差限 R 为

```
y =                    R =
    0.6434                8.8610e - 004
```

方法 2(求牛顿插值多项式和差商的 MATLAB 主程序) 输入 MATLAB 程序

```
>> x = 2 * pi/9; X = [pi/6 ,pi/4, pi/3];
Y = [0.5,0.7071,0.8660]; M = 1;
[A,C,L,wcgs,Cw] = newpoly (X,Y), y = polyval(C,x)
```

运行后输出结果

```
A =
    0.5000         0          0
    0.7071    0.7911          0
    0.8660    0.6070    -0.3516
C =
     -0.3516   1.2513   -0.0588
L =
       - 1583578379808357/ 4503599627370496 * x ^ 2 + 1408883391907005/
112.[illegible]            41.91[illegible]5248
wcgs =
```

```
    1/6*M*(x^3-3/4*x^2*pi+4012734077357799/2251799813685248
*x-7757769783530263/18014398509481984)
    Cw =
        0.1667   -0.3927   0.2970   -0.0718
    y =
        0.6434
```

上述两种方法计算 y 的结果相同.

请读者求函数 $f(x)=e^{-3x}$ 在 $[0,4]$ 上的牛顿插值多项式,并估计其误差.

6.3.5 牛顿插值法的 MATLAB 综合程序

在例 6.3.4 中,分两次分别求出牛顿插值多项式、差商、插值及其误差估计,比较麻烦. 如果我们想同时求出牛顿插值多项式、差商、插值及其误差估计,那么请用下面编写的名为 newdscg.m 的 M 文件,这套 MATLAB 主程序能快捷地达到您的目的.

求牛顿插值多项式、差商、插值及其误差估计的 MATLAB 主程序

输入的量:X 是 $n+1$ 个节点 (x_i, y_i) $(i=1,2,\cdots,n+1)$ 横坐标向量,Y 是纵坐标向量,x 是以向量形式输入的 m 个插值点,M 在 $[a,b]$ 上满足 $|f^{(n+1)}(x)|\leq M$.

输出的量:向量 y 是向量 x 处的插值,误差限 R,n 次牛顿插值多项式 L 及其系数向量 C,差商的矩阵 A.

```
function [y,R,A,C,L] = newdscg(X,Y,x,M)
n = length(X); m = length(x);
for t = 1:m
    z = x(t); A = zeros(n,n);A(:,1) = Y';
    s = 0.0; p = 1.0; q1 = 1.0; c1 = 1.0;
          for j = 2:n
              for i = j:n
              A(i,j) = (A(i,j-1) - A(i-1,j-1))/(X(i) -X(i-j+1));
              end
                q1 = abs(q1*(z-X(j-1)));c1 = c1*j;
          end
            C = A(n,n);q1 = abs(q1*(z-X(n)));
            for k = (n-1):-1:1
              C = conv(C,poly(X(k)));
              d = length(C);C(d) = C(d) +A(k,k);
            end
```

```
        y(k) = polyval(C, z);
    end
        R = M * q1 / c1;L(k,:) = poly2sym(C);
```

例 6.3.6 给出节点数据 $f(-4.00)=27.00$, $f(0.00)=1.00$, $f(1.00)=2.00$, $f(2.00)=17.00$ 作三阶牛顿插值多项式,计算 $f(-2.345)$,并估计其误差.

解 首先将名为 newdscg.m 的程序保存为 M 文件,然后在 MATLAB 工作窗口输入程序

```
>> syms M,X = [ -4,0,1,2]; Y = [27,1,2,17]; x = -2.345;
[y,R,A,C,P] = newdscg(X,Y,x,M)
```

运行后输出插值 $y \approx f(-2.345)$ 及其误差限公式 $\boldsymbol{R}$,三阶牛顿插值多项式 $\boldsymbol{P}$ 及其系数向量 $\boldsymbol{C}$,差商的矩阵 $\boldsymbol{A}$ 如下

```
y =
    22.3211
R =
    1323077530165133/562949953421312 * M(即 R =2.3503 * M)
A =
    27.0000          0         0         0
     1.0000    -6.5000         0         0
     2.0000     1.0000    1.5000         0
    17.0000    15.0000    7.0000    0.9167
C =
    0.9167   4.2500   -4.1667   1.0000
P =
    11/12 * x^3 +17/4 * x^2 -25/6 * x +1
```

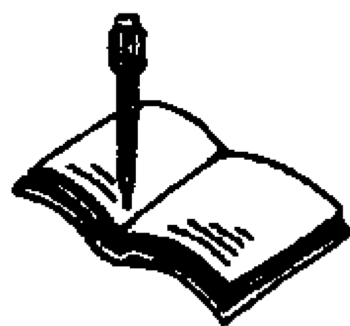
例 6.3.7 将区间 $[0,\pi/2]$ 分成 n 等份($n=2,3$),用 $y=f(x)=\sin x$ 产生 $n+1$ 个节点,求二阶和三阶牛顿插值多项式 $P_2(x)$ 和 $P_3(x)$. 用它们分别计算 $\sin(\pi/7)$(取四位有效数字),并估计其误差.

解 首先将名为 newdscg.m 的程序保存为 M 文件,然后在 MATLAB 工作窗口输入程序

```
>> X1 = 0:pi/4:pi/2; Y1 = sin(X1); M = 1; x = pi/7; X2 = 0:pi/6:pi/
2; Y2 = sin(X2);
[y1,R1,A1,C1,P2] = newdscg(X1,Y1,x,M),
[y2,R2,A2,C2,P3] - newdscg(X2,Y2,x,M)
```

运行后输出插值 $y_1=P_2(\pi/7)\approx\sin(\pi/7)$ 和 $y_2=P_3(\pi/7)\approx\sin(\pi/7)$ 及其误差限 R_1 和 R_2,二阶和三阶牛顿插值多项式 P_2 和 P_3 及其系数向量 $\boldsymbol{C}_1$ 和 $\boldsymbol{C}_2$,差商的矩阵 $\boldsymbol{A}_1$ 和 $\boldsymbol{A}_2$ 如下

```
y1 =
    0.4548
R1 =
    0.0282
A1 =
         0         0         0
    0.7071    0.9003         0
    1.0000    0.3729   -0.3357
C1 =
   -0.3357    1.1640    0
P2 =
   -3024156947890437/9007199254740992*x^2+163820246322191/
140737488355328*x
y2 =
    0.4345
R2 =
    9.3913e-004
A2 =
         0         0         0         0
    0.5000    0.9549         0         0
    0.8660    0.6991   -0.2443         0
    1.0000    0.2559   -0.4232   -0.1139
C2 =
   -0.1139   -0.0655    1.0204    0
P3 =
 -1025666884451963/9007199254740992*x^3-4717668559161127/
72057594037927936*x^2+4595602396951547/4503599627370496*x
```



习 题 6.3

1. 已知函数$f(x)$在$[1,7]$上具有二阶连续导数，$|f''(x)|\leqslant 5$，且满足条件$f(1)=1$，$f(7)=12$. 求一阶牛顿插值多项式和函数值$f(3.5)$，并估计其误差.

2. 求函数$f(x)=e^{-3x}$在$[0,4]$上的六阶牛顿插值多项式和估计误差的公式.

3. 将区间$[\pi/6,\pi/2]$分成n等份$(n=1,2)$，用$y=f(x)=\sin x$产生$n+1$个节点，求二阶和三阶牛顿插值多项式$P_2(x)$和$P_3(x)$. 用它们分别计算$\sin(\pi/7)$（取四位有效数字），并估计其误差.

4. 给出节点数据$f(-3.00)=27.00$，$f(0.00)=1.00$，$f(1.00)=2.00$，$f(2.00)=17.00$

作三阶牛顿插值多项式计算$f(1.4)$,并估计其误差.

5. 给出节点数据$f(-3.15)=37.03$, $f(-1.00)=7.24$, $f(0.01)=1.05$, $f(1.02)=2.03$, $f(2.03)=17.06$, $f(3.25)=23.05$作五阶牛顿插值多项式和差商,并写出估计其误差的公式.

6. 已知$\sin 30°=0.5$, $\sin 45°=0.7071$, $\sin 90°=1$,用牛顿插值法求$\sin 40°$的近似值,并估计其误差.

6.4 埃尔米特(Hermite)插值及其MATLAB程序

在对函数$f(x)$构造插值函数时,有时不仅要求在节点处插值多项式$P(x)$的值等于$f(x)$的值,还要求在节点处插值多项式$P(x)$的导数$P'(x)$的值等于$f(x)$的导数$f'(x)$的值,这样的插值称**埃尔米特插值**. 本节给出埃尔米特插值多项式和误差估计及其MATLAB程序.

6.4.1 埃尔米特插值多项式

埃尔米特插值多项式可以满足节点处函数、一阶导数均与给定值相等的要求. 由于给定值增加了一倍,插值多项式的待定系数也要增加一倍,于是当节点数为$n+1$时,可唯一确定一个次数不超过$2n+1$的插值多项式.

定义6.2 设函数$f(x)$在$[a,b]$上有一阶连续导数,且$n+1$个互异点$x_0, x_1, x_2, \cdots, x_n \in [a,b]$. 如果存在至多为$2n+1$阶的多项式$H_{2n+1}(x)$满足

$$\begin{cases} H_{2n+1}(x_j)=f(x_j), \\ H'_{2n+1}(x_j)=f'(x_j) \end{cases} \quad (j=0,1,2,\cdots,n), \tag{6.32}$$

则称$H_{2n+1}(x)$为函数$f(x)$在点$x_0, x_1, x_2, \cdots, x_n$的$2n+1$阶**埃尔米特插值多项式**.

给定函数$f(x)$在$n+1$个互异点$x_0, x_1, x_2, \cdots, x_n$处的函数值$f(x_j)=y_j(j=0,1,\cdots,n)$和导数值$f'(x_j)=y'_j(j=0,1,\cdots,n)$,下面求函数$f(x)$在点$x_0, x_1, x_2, \cdots, x_n$处的$2n+1$阶埃尔米特插值多项式$H_{2n+1}(x)$,使其满足(6.32)式.

我们可以借鉴拉格朗日插值法构造基函数的思想,设埃尔米特插值多项式$H_{2n+1}(x)$为

$$H_{2n+1}(x) = \sum_{i=0}^{n} H_i(x)y_i + \sum_{i=0}^{n} h_i(x)y'_i, \tag{6.33}$$

其中$H_i(x)$和$h_i(x)$都是$2n+1$阶多项式. 且满足

$$\begin{cases} H_i(x_j)=\delta_{ij}=\begin{cases} 0, j\neq i, \\ 1, j=i, \end{cases} \quad (i=0,1,2,\cdots,n), \\ H'_i(x_j)=0 \end{cases} \tag{6.34}$$

$$\begin{cases} h_i'(x_j)=\delta_{ij}=\begin{cases}0, j\neq i,\\ 1, j=i,\end{cases} & (i=0,1,2,\cdots,n).\\ h_i(x_j)=0 \end{cases} \tag{6.35}$$

先求 $H_i(x)$，根据 $H_i(x_j)=\delta_{ij}$ 可知，拉格朗日插值的基函数 $l_i(x)$ 的零因子都是 $H_i(x)$ 的零因子. 又因为 $H_i'(x_j)=0$，所以 $H_i(x)$ 的零因子都是二重零因子，即 $H_i(x)$ 含有 $l_i^2(x)$ 因子，显然 $l_i^2(x)$ 是 $2n$ 次多项式. 因此设

$$H_i(x)=(ax+b)l_i^2(x). \tag{6.36}$$

将(6.34)式的 $H_i(x_i)=1$ 和 $l_i(x_i)=1$ 代入(6.36)式，得

$$ax_i+b=1. \tag{6.37}$$

由(6.36)式得

$$H_i'(x)=al_i^2(x)+2(ax+b)l_i(x)l_i'(x). \tag{6.38}$$

将(6.35)式的 $H_i'(x_i)=0$ 代入(6.38)式，得到的式子与(6.37)式联立解得

$$\begin{cases} a=-2l_i'(x_i),\\ b=1+2x_il_i'(x_i). \end{cases} \tag{6.39}$$

将(6.39)式代入(6.36)式，得到

$$H_i(x)=[1+2x_il_i'(x_i)-2l_i'(x_i)\cdot x]l_i^2(x),$$

$$H_i(x)=[1-2(x-x_i)l_i'(x_i)]l_i^2(x)$$

$$=\left[1-2(x-x_i)\sum_{\substack{j=0\\ j\neq i}}^{n}\frac{1}{x_i-x_j}\right]l_i^2(x). \tag{6.40}$$

同理可得

$$h_i(x)=(x-x_i)l_i^2(x). \tag{6.41}$$

将(6.40)和(6.41)式代入(6.33)式，得到

$$H_{2n+1}(x)=\sum_{i=0}^{n}[1-2(x-x_i)l_i'(x_i)]l_i^2(x)y_i+\sum_{i=0}^{n}(x-x_i)l_i^2(x)y_i'.$$

满足条件(6.32)的 $2n+1$ 阶的插值多项式 $H_{2n+1}(x)$ 是唯一的. 事实上，设另有一个多项式 $2n+1$ 阶的插值多项式 $G_{2n+1}(x)$ 也满足条件(6.32). 令

$$q(x)=H_{2n+1}(x)-G_{2n+1}(x).$$

则点 $x_0,x_1,x_2,\cdots,x_n$ 都是 $q(x)$ 的二重根，从而 $q(x)$ 至少有 $2n+2$ 个根. 但是不高于 $2n+1$ 阶的多项式 $q(x)$ 至多有 $2n+1$ 个根，因此，$q(x)$ 只能是零次多项式.

综上所述，我们得到下面的定理.

定理 6.5 如果函数 $f(x)$ 在 $n+1$ 个互异点 $x_0,x_1,x_2,\cdots,x_n$ 处存在函数值

$f(x_j)=y_j(j=0,1,\cdots,n)$ 和导数值 $f'(x_j)=y_j'(j=0,1,\cdots,n)$，则函数 $f(x)$ 在点 $x_0,x_1,x_2,\cdots,x_n$ 处存在唯一一个 $2n+1$ 阶**埃尔米特插值多项式**

$$H_{2n+1}(x)=\sum_{i=0}^{n}[1-2(x-x_i)l_i'(x_i)]l_i^2(x)y_i+\sum_{i=0}^{n}(x-x_i)l_i^2(x)y_i',\tag{6.42}$$

其中 n 次多项式

$$l_i(x)=\frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$$

$$=\sum_{\substack{j=0\\j\neq i}}^{n}\frac{x-x_j}{x_i-x_j},i=0,1,\cdots,n,$$

$$l_i'(x_i)=\sum_{\substack{j=0\\j\neq i}}^{n}\frac{1}{x_i-x_j},i=0,1,\cdots,n.$$

特别重要的是，当 $n=1$ 时，满足 $H_3(x_0)=y_0,H_3(x_1)=y_1,H_3'(x_0)=y_0'$，$H_3'(x_1)=y_1'$ 的**三阶埃尔米特插值多项式为**

$$H_3(x)=\left[\left(1+2\frac{x-x_0}{x_1-x_0}\right)y_0+(x-x_0)y_0'\right]\left(\frac{x-x_1}{x_0-x_1}\right)^2+$$

$$\left[\left(1+2\frac{x-x_1}{x_0-x_1}\right)y_1+(x-x_1)y_1'\right]\left(\frac{x-x_0}{x_1-x_0}\right)^2\tag{6.43}$$

例 6.4.1 给定函数 $f(x)$ 的两个函数值 $f(-1)=0,f(1)=4$ 和两个导数值 $f'(-1)=2,f'(1)=0$，求函数 $f(x)$ 在点 $x_0=-1,x_1=1$ 处的三阶埃尔米特插值多项式 $H_3(x)$，并计算 $f(0.5)$.

解 将 $x_0=-1,x_1=1,y_0=0,y_1=4,y_0'=2,y_1'=0$ 代入(6.43)，得

$$H_3(x)=\left[\left(1+2\frac{x-(-1)}{1-(-1)}\right)0+(x-(-1))2\right]\left(\frac{x-1}{-1-1}\right)^2+$$

$$\left[\left(1+2\frac{x-1}{-1-1}\right)4+(x-1)0\right]\left(\frac{x-(-1)}{1-(-1)}\right)^2.$$

即
$$H_3(x)=\frac{1}{2}(x+1)\cdot(x-1)^2+(x+1)^2\cdot(2-x),$$

则
$$H_3(0.5)=3.5625.$$

6.4.2 误差估计

定理 6.6 如果 $H_{2n+1}(x)$ 是函数 $f(x)$ 在点 $x_0,x_1,x_2,\cdots,x_n$ 的 $2n+1$ 阶埃尔米特插值多项式，$f(x)$ 在含点 $x_0,x_1,x_2,\cdots,x_n$ 的区间 $[a,b]$ 上具有 $2n+2$ 阶导数，则对于任意 $x\in[a,b]$，总存在 $\xi\in(a,b)$，使得

$$R_{2n+1}(x)=f(x)-H_{2n+1}(x)=\frac{f^{(2n+2)}(\xi)}{(2n+2)!}\omega^2(x),\xi\in(a,b),\qquad(6.44)$$

其中 $\omega(x)=(x-x_n)\cdots(x-x_1)(x-x_0)$.

特别地,当 $n=1$ 时,三阶埃尔米特插值多项式(6.43)的误差为

$$R_3(x)=f(x)-H_3(x)=\frac{f^{(4)}(\xi)}{4!}(x-x_0)^2(x-x_1)^2,\xi \text{ 在 } x_0 \text{ 和 } x_1 \text{ 之间}.$$

证明 (1) 当 $x=x_i(i=0,1,2,\cdots,n)$时,$f(x_i)=H_{2n+1}(x_i)$,$\omega(x_i)=0$ $(i=0,1,\cdots,n)$,(6.44)式成立.

(2) 当 $x\neq x_i(i=0,1,2,\cdots,n)$时,作一个辅助函数

$$g(t)=f(t)-H_{2n+1}(t)-\frac{f(x)-H_{2n+1}(x)}{\omega^2(x)}\omega^2(t).$$

因为 $g(x_i)=0$ $(i=0,1,\cdots,n)$,$g(x)=0$ $(x\neq x_i,i=0,1,2,\cdots,n)$,即 $g(t)$ 有 $n+2$ 个零点,由罗尔定理知,至少存在 $n+1$ 个 ξ_j,且 $\xi_j\neq x_i$,$\xi_j\neq x$ $(j=0,1,2,\cdots,n)$使得 $g'(\xi_j)=0$. 又因为 $g'(x_i)=0$ $(i=0,1,2,\cdots,n)$,所以 $g'(t)$至少有 $2n+2$ 个互异的零点. 对 $g'(t)$反复用罗尔定理 $2n+1$ 次,则至少存在一点 ξ,使得 $g^{(2n+2)}(\xi)=0$. 即

$$g^{(2n+2)}(\xi)=f^{(2n+2)}(\xi)-\frac{f(x)-H_{2n+2}(x)}{\omega^2(x)}(2n+2)!\ =0.$$

因此,$R_{2n+1}(x)=f(x)-H_{2n+1}(x)=\dfrac{f^{(2n+2)}(\xi)}{(2n+2)!}\omega^2(x)$,$\xi\in(a,b)$. 证毕

例 6.4.2 给定函数 $f(x)$在 3 个互异点 x_0,x_1,x_2 处的函数值 $f(x_j)=y_j(j=0,1,2)$和导数值 $f'(x_j)=y_j'(j=0,1,2)$,求函数 $f(x)$在点 x_0,x_1,x_2 处的五阶埃尔米特插值多项式 $H_5(x)$和误差公式,使其满足

$$\begin{cases}H_5(x_j)=f(x_j),\\ H_5'(x_j)=f'(x_j)\end{cases}\quad(j=0,1,2).$$

解 根据定理 6.5 知,函数 $f(x)$在点 x_0,x_1,x_2 处存在唯一一个五阶埃尔米特插值多项式

$$H_5(x)=\sum_{i=0}^{2}\{[1-2(x-x_i)l_i'(x_i)]y_i+(x-x_i)y_i'\}l_i^2(x),$$

其中二次多项式为

$$l_0(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)},l_1(x)=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)},l_2(x)=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}.$$

$$l_0'(x_0)=\frac{1}{x_0-x_1}+\frac{1}{x_0-x_2},l_1'(x_1)=\frac{1}{x_1-x_0}+\frac{1}{x_1-x_2},l_2'(x_2)=\frac{1}{x_2-x_0}+\frac{1}{x_2-x_1},$$

$$R(x)=f(x)-H_5(x)=\frac{f^{(6)}(\xi)}{6!}(x-x_0)^2(x-x_1)^2(x-x_2)^2,\xi \text{ 在点 } x_0,x_1,x_2 \text{ 之间}.$$

6.4.3 埃尔米特插值多项式和误差估计的MATLAB程序

如果函数$f(x)$在$n+1$个互异点$x_0,x_1,x_2,\cdots,x_n$处存在函数值$f(x_j)=y_j(j=0,1,\cdots,n)$和导数值$f'(x_j)=y'_j(j=0,1,\cdots,n)$，则函数$f(x)$在点$x_0,x_1,x_2,\cdots,x_n$处$2n+1$阶埃尔米特插值多项式和误差公式可以用下面编写的MATLAB主程序计算.

求埃尔米特插值多项式和误差估计的MATLAB主程序

输入的量：$n+1$个节点$(x_i,f(x_i))$ $(i=1,2,\cdots,n+1)$横坐标向量$\boldsymbol{X}$，纵坐标向量$\boldsymbol{Y}$，以$f'(x_i)=y'_i(i=1,2,\cdots,n+1)$为元素的向量$\boldsymbol{Y}_1$；

输出的量：$2n+1$阶埃尔米特插值多项式H_k及其系数向量$\boldsymbol{H}_c$，误差公式$wcgs$及其系数向量$\boldsymbol{C}_w$.

```
function [Hc, Hk,wcgs,Cw] = hermite (X,Y,Y1)
m = length(X); n = M1;s = 0; H = 0;q = 1;c1 = 1; L = ones(m,m); G = ones(1,m);
for k = 1:n +1
    V = 1;
    for i = 1:n +1
        if k ~ = i
            s = s + (1 /(X(k) - X(i)));
            V = conv(V,poly(X(i))) /(X(k) - X(i));
        end
          h = poly(X(k)); g = (1 - 2 * h * s); G = g * Y(k) + h * Y1(k);
    end
      Hc = H + conv(G,conv(V,V)); b = poly(X(k));b2 = conv(b,b);
      q = conv(q,b2); t = 2 * n + 2;
      Hc - H;Hk = poly2sym (H); Q = poly2sym(q);
end
for i = 1:t
    c1 = c1 * i;
end
syms M,wcgs = M * Q /c1; Cw = q /c1;
```

例6.4.3 给定函数$f(x)$在点$x_0=\pi/6,x_1=\pi/4,x_2=\pi/2$处的函数值$f(x_0)=0.5,f(x_1)=0.707\ 1,f(x_2)=1$和导数值$f'(x_0)=0.866\ 0,f'(x_1)=0.707\ 1,f'(x_0)=0.000\ 0$，且$|f^{(6)}(x)|\leqslant 1$，求函数$f(x)$在点$x_0,x_1,x_2$处的五阶埃尔米特插值多项式$H_5(x)$和误差公式，计算$f(1.567)$并估计其误差.

解 (1)保存名为`hermite.m`的M文件.

(2)在MATLAB工作窗口输入程序

```
>>X = [pi/6,pi/4,pi/2]; Y = [0.5,0.7071,1];
Y1 = [0.8660,0.7071,0]; [Hc, Hk,wcgs,Cw] = hermite (X,Y,Y1)
```

运行后输出五阶埃尔米特插值多项式 H_k 及其系数向量 H_c，误差公式 *wcgs* 及其系数向量 C_w 如下

```
Hc =
  1.0e+003 *
    0.1912   -0.9273   1.6903   -1.4380   0.5751   -0.0866
Hk =
6725828781679091/35184372088832*x^5-4078286086775209/4398046511104*x^4+7434035571017927/4398046511104*x^3-3162205449085973/2199023255552*x^2+5058863928652835/8796093022208*x-6094057839958843/70368744177664
wcgs =
1/720*M*(x^6-11/6*x^5*pi+7446708432019761/562949953421312*x^4-4363745503235773/281474976710656*x^3+21569239021155/2199023255552*x^2-7178073637328281/2251799813685248*x+3758430567659515/9007199254740992)
Cw =
 0.0014  -0.0080  0.0184  -0.0215  0.0136  -0.0044  0.0006
```

当 $|f^{(6)}(x)| \leqslant 1 = M$ 时的误差公式为

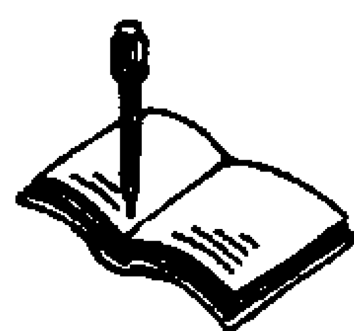
R = 0.001 4 * x^6 - 0.008 0 * x^5 + 0.018 4 * x^4 - 0.021 5 * x^3 + 0.013 6 * x^2 - 0.004 4 * x + 0.000 6

(3) 在 MATLAB 工作窗口输入程序

```
>>x = 1.567;M = 1;
Hk = 6725828781679091/35184372088832 * x^5 - 4078286086775209/4398046511104*x^4 +7434035571017927/4398046511104 * x^3 -3162205449085973/2199023255552 * x^2 + 5058863928652835/8796093022208 * x - 6094057839958843/70368744177664,
wcgs = 1/720 * M * (x^6 - 11/6 * x^5 * pi + 7446708432019761/562949953421312*x^4  4363745503235773/281474976710656 * x^3 +21569239021155/2199023255552 * x^2 - 7178073637328281/2251799813685248 * x + 3758430567659515/9007199254740992)
```

运行后输出 $f(1.567)$ 的近似值 H_k 及其误差 *wcgs* 如下

```
Hk =
    2.5265
wcgs =
    1.3313e-008
```



习 题 6.4

1. 给定函数$f(x)$在点$x_0=\pi/6, x_1=\pi/4$处的函数值$f(x_0)=0.5, f(x_1)=0.707\ 1$和导数值$f'(x_0)=0.866\ 0, f'(x_1)=0.707\ 1$，且$|f^{(4)}(x)|\leqslant 1$，求函数$f(x)$在点$x_0, x_1$处的三阶埃尔米特插值多项式$H_3(x)$和误差公式.

2. 求函数$f(x)=e^{-3x}$在$[0,4]$上的五阶埃尔米特插值多项式，并估计其误差.

3. 将区间$[\pi/6, \pi/2]$分成n等份$(n=1,2)$，用$y=f(x)=\sin x$产生$n+1$个节点，然后根据(6.42)和(6.44)式分别作埃尔米特插值多项式及其误差公式. 用它们分别计算$\sin(\pi/5)$（取四位有效数字），并估计其误差.

4. 给出节点数据$f(3.00)=27.00, f(1.00)=1.00, f'(1.00)=2.00, f'(3.00)=17.00$作埃尔米特插值多项式，计算$f(1.4)$，并估计其误差.

5. 给出节点数据$f(-3.15)=37.03, f(-1.00)=7.24, f(0.01)=1.05, f'(-3.15)=2.03, f'(-1.00)=17.06, f'(0.01)=23.05$作埃尔米特插值多项式，并写出估计误差的公式.

6. 已知$\sin 30°=0.5, \sin 45°=0.707\ 1, \sin 90°=1, f'(x_0)=0.866\ 0, f'(x_1)=0.707\ 1, f'(x_0)=0.000\ 0$，且$|f^{(6)}(x)|\leqslant 1$，作埃尔米特插值多项式，求$\sin 40°$的近似值，并估计其误差.

6.5 分段插值及其 MATLAB 程序

本节首先用 MATLAB 程序作出两个已知函数的$n=2,4,6,8,10$次拉格朗日插值多项式$L_n(x)$的图形，用以说明当$n\to\infty$时，在$[a,b]$内并不能保证$L_n(x)$处处收敛于$f(x)$. 然后给出克服这些问题的方法，分段插值和分段线性插值及其 MATLAB 程序.

6.5.1 高次插值的振荡

在拉格朗日插值方法中，为了在插值区间$[a,b]$上使插值多项式$L_n(x)$更好地逼近被插值函数$f(x)$，往往要增加插值节点，提高插值多项式$L_n(x)$的次数. 虽然在大多数情况下，随着节点个数的增加，$L_n(x)$的次数n变大，误差的绝对值$|R_n(x)|$会变小. 但是在某些情况下，过分地提高插值多项式的次数会带来一些新的问题. 例如，随着节点个数n的增加，$L_n(x)$的光滑性变坏，有时会出现很大的振荡. 理论上，当$n\to\infty$时，在$[a,b]$内并不能保证$L_n(x)$处处收敛于$f(x)$. 下面我们给出两个例子说明高次插值的振荡现象.

例 6.5.1 作下列函数在指定区间上的n次拉格朗日插值多项式$L_n(x)$ $(n=2,4,6,8,10)$的图形，并讨论插值多项式$L_n(x)$的次数与误差$R_n(x)$的关系.

(1) $f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right), x\in[-\pi,\pi]$；(2) $f(x)=\frac{1}{1+x^2}, x\in[-5,5]$.

解 (1)① 将区间$[-\pi,\pi]$进行 n ($n=2,4,6,8,10$)等分，过 $n+1$ 个等分点作函数$f(x)$的 n 次拉格朗日插值多项式 $L_n(x)$.

② 将下面计算 n 次拉格朗日插值多项式 $L_n(x)$的值的 MATLAB 程序保存名为 lagr1.m 的 M 文件

```
function y = lagr1(x0,y0,x)
n = length(x0); m = length(x);
for i =1:m
  z = x(i);s = 0.0;
for k =1:n
      p =1.0;
  for j =1:n
    if j ~ =k
          p =p*(z -x0(j))/(x0(k) -x0(j));
    end
  end
  s =p*y0(k) +s;
end
y(i) =s;
end
```

③ 现提供作 n 次拉格朗日插值多项式 $L_n(x)$的图形的 MATLAB 程序，保存名为 lil.m 的 M 文件

```
m =150; x = -pi:2*pi/(m -1): pi;
y = tan(cos((3^(1/2) +sin(2*x))./(3 +4*x.^2)));
plot(x,y,'k -'),
gtext('y =tan(cos((sqrt(3) +sin(2x))/(3 +4x^2)))'),pause
n =3; x0 = -pi:2*pi/(3 -1):pi;
y0 =tan(cos((3^(1/2) +sin(2*x0))./(3 +4*x0.^2)));
y1 =lagr1(x0,y0,x);hold on,
plot(x,y1,'g <'),gtext('n =2 '),pause,hold off
n =5; x0 = -pi:2*pi/(n -1):pi;
y0 =tan(cos((3^(1/2) +sin(2*x0))./(3 +4*x0.^2)));
y2 =lagr1(x0,y0,x);hold on,
plot(x,y2,'b:'),gtext('n =4 '),pause,hold off
n =7; x0 = -pi:2*pi/(n -1):pi;
y0 =tan(cos((3^(1/2) +sin(2*x0))./(3 +4*x0.^2)));
y3 =lagr1(x0,y0,x);hold on,
```

```
plot(x,y3,'rp'),gtext('n=6'),pause,hold off
n=9; x0=-pi:2*pi/(n-1):pi;
y0=tan(cos((3^(1/2)+sin(2*x0))./(3+4*x0.^2)));
y4=lagr1(x0,y0,x);hold on,
plot(x,y4,'m*'),gtext('n=8'),pause,hold off
n=11; x0=-pi:2*pi/(n-1):pi;
y0=tan(cos((3^(1/2)+sin(2*x0))./(3+4*x0.^2)));
y5=lagr1(x0,y0,x);hold on,
plot(x,y5,'g:'),gtext('n=10')
title('高次拉格朗日插值的振荡')
```

在 MATLAB 工作窗口输入名为 `li1.m` 的 M 文件的文件名

```
>>li1.m
```

回车运行后，便会逐次画出 $f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$ 在区间 $[-\pi,\pi]$ 上的 n 次拉格朗日插值多项式 $L_n(x)$ $(n=2,4,6,8,10)$ 的图形（见图 6-1(a)）.

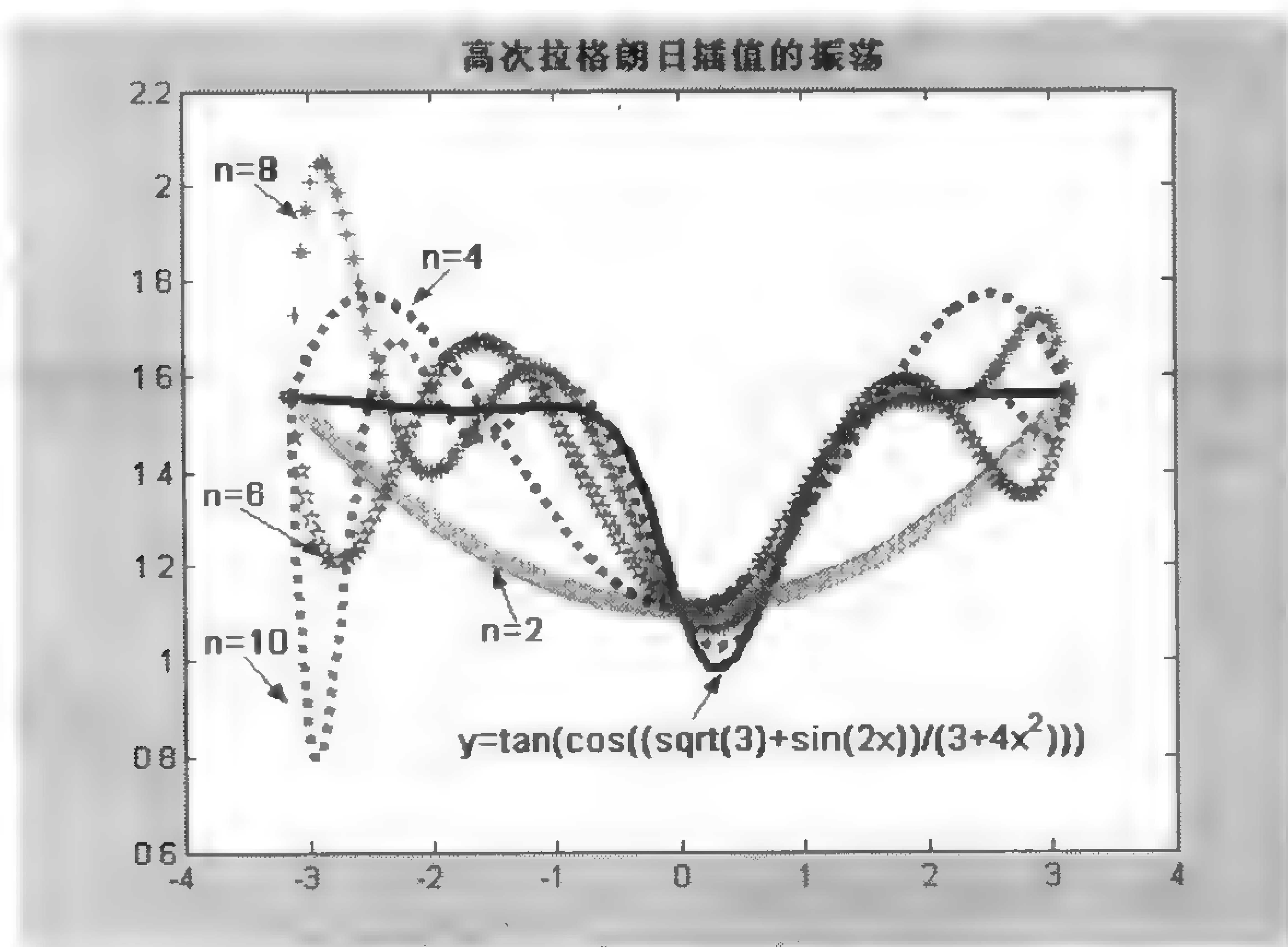


图 6-1(a)　$f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$ 的拉格朗日插值曲线 $L_n(x)$

(2) 在 MATLAB 工作窗口输入程序

```
m=101; x=-5:10/(m-1):5; y=1./(1+x.^2);
z=0*x;plot(x,z,'r',x,y,'k:'),
gtext('y=1/(1+x^2)'),pause
```

```
n=3; x0 = -5:10/(n-1):5; y0 =1./(1+x0.^2);
y1 = lagr1(x0,y0,x);hold on,
plot(x,y1,'g'),gtext('n=2'),pause,hold off
n=5; x0 = -5:10/(n-1):5; y0 =1./(1+x0.^2);
y2 = lagr1(x0,y0,x);hold on,
plot(x,y2,'b:'),gtext('n=4'),pause,hold off
n=7; x0 = -5:10/(n-1):5; y0 =1./(1+x0.^2);
y3 = lagr1(x0,y0,x);hold on,
plot(x,y3,'r'),gtext('n=6'),pause,hold off
n=9; x0 = -5:10/(n-1):5; y0 =1./(1+x0.^2);
y4 = lagr1(x0,y0,x);hold on,
plot(x,y4,'r:'),gtext('n=8'),pause,hold off
n-11; x0 = -5:10/(n-1):5; y0 =1./(1+x0.^2);
y5 = lagr1(x0,y0,x);hold on,
plot(x,y5,'m'),gtext('n=10')
title('高次插值多项式的振荡')
```

回车运行后,便会逐次画出 $y=\dfrac{1}{1+x^2}$ 在区间$[-5,5]$上的 n 次拉格朗日插值多项式 $L_n(x)$ $(n=2,4,6,8,10)$的图形,见图 6-1(b).

从图 6-1 可以看出,对于较大的$|x|$,随着 n 的增大,$L_n(x)$振荡越来越大.事实上可以证明,仅当 $|x|\leqslant\xi$ (ξ 是某一个正数)时,才有 $\lim\limits_{n\to\infty}L_n(x)=g(x)$. 而在此区间外,$L_n(x)$是发散的.由于在大范围内使用高次插值多项式逼近的效果往往不理想,从而促使人们转而寻求简单的低次插值方法.

6.5.2 分段插值和分段线性插值

分段插值法可以克服在大范围内使用高次插值所带来的问题.所谓的**分段插值法**就是首先在插值区间$[a,b]$上插入点

$$a=x_0<x_1<x_2<\cdots<x_n=b,$$

然后在每个小的子区间$[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$上构造低次插值多项式 $l_i(x)$,再将每个子区间$[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$上多项式 $l_i(x)$连接,作为插值区间$[a,b]$上的插值函数 $P(x)$.

分段插值法是一种显式算法,不但算法简单,而且具有良好的收敛性.只要每两个相邻节点之间的间距充分小,分段插值法总能获得所要求的精度,而不会出现高次插值那样的振荡.另外分段插值具有局部性,修改某个数据,插值曲线仅仅在某个局部范围内受到影响,而代数插值却影响到整个插值区间.

分段插值法中最简单的方法是分段线性插值.分段线性插值函数是将每两

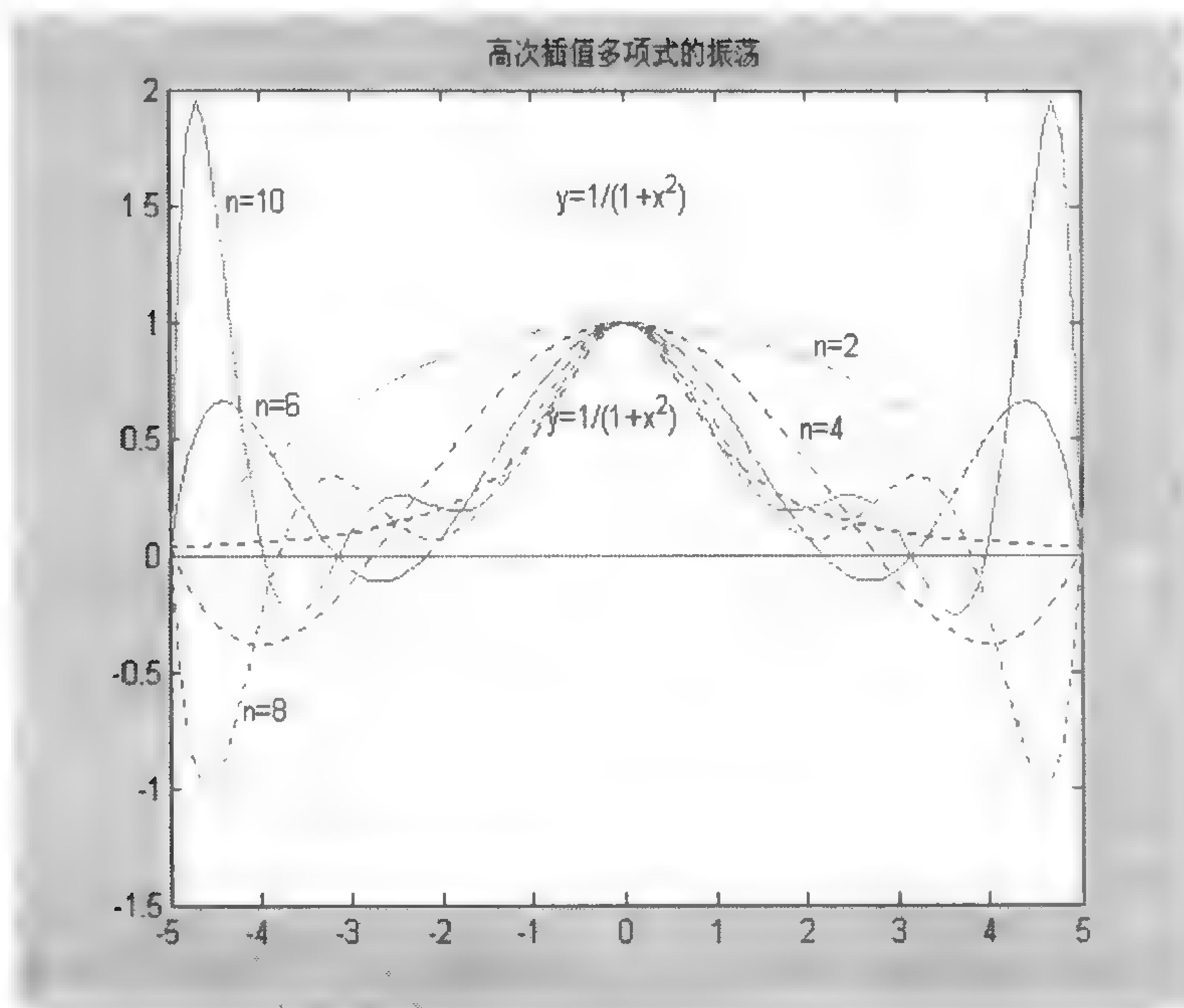


图 6-1(b) $y=1/(1+x^2)$的拉格朗日插值曲线 $L_n(x)$

个相邻的节点用直线段连起来,形成一条折线. 例如,图 6-2 是函数 $y=\cos x$ 在节点(x_i,y_i) (其中 $x_i=-6+1.5i$ $(i=0,1,2,\cdots,12)$)处的分段线性插值函数和节点的图形,蓝色的小圆表示节点(x_i,y_i),绿色的实线表示分段线性插值函数. 下面给出分段线性插值函数的严格定义.

定义 6.3 设函数$f(x)$在$[a,b]$上的 $n+1$ 个点 $a=x_0<x_1<x_2<\cdots<x_n=b$ 处的函数值为 $f(x_i)=y_i(i=0,1,2,\cdots,n)$. 连接每两个相邻的节点$(x_i,y_i)$和$(x_{i+1},y_{i+1})$,作一条折线函数 $S_n(x)$,使得 $S_n(x)$满足如下条件:

(1) $S_n(x)$在$[a,b]$上连续;

(2) $S_n(x_i)=y_i(i=0,1,2,\cdots,n)$;

(3) $S_n(x)$在每个子区间$[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$上是线性函数 $l_i(x)$,

则称折线函数 $S_n(x)$为**分段线性插值函数**.

定理 6.7 如果 $S_n(x)$是定义 6.3 所定义的分段线性插值函数,则有下面结论:

(1) $S_n(x)$在每个子区间$[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$上形式如下

$$S_i(x)=\frac{x-x_i}{x_{i-1}-x_i}y_{i-1}+\frac{x-x_{i-1}}{x_i-x_{i-1}}y_i \quad (x_{i-1}\leqslant x\leqslant x_i, i=1,2,\cdots,n), \qquad (6.45)$$

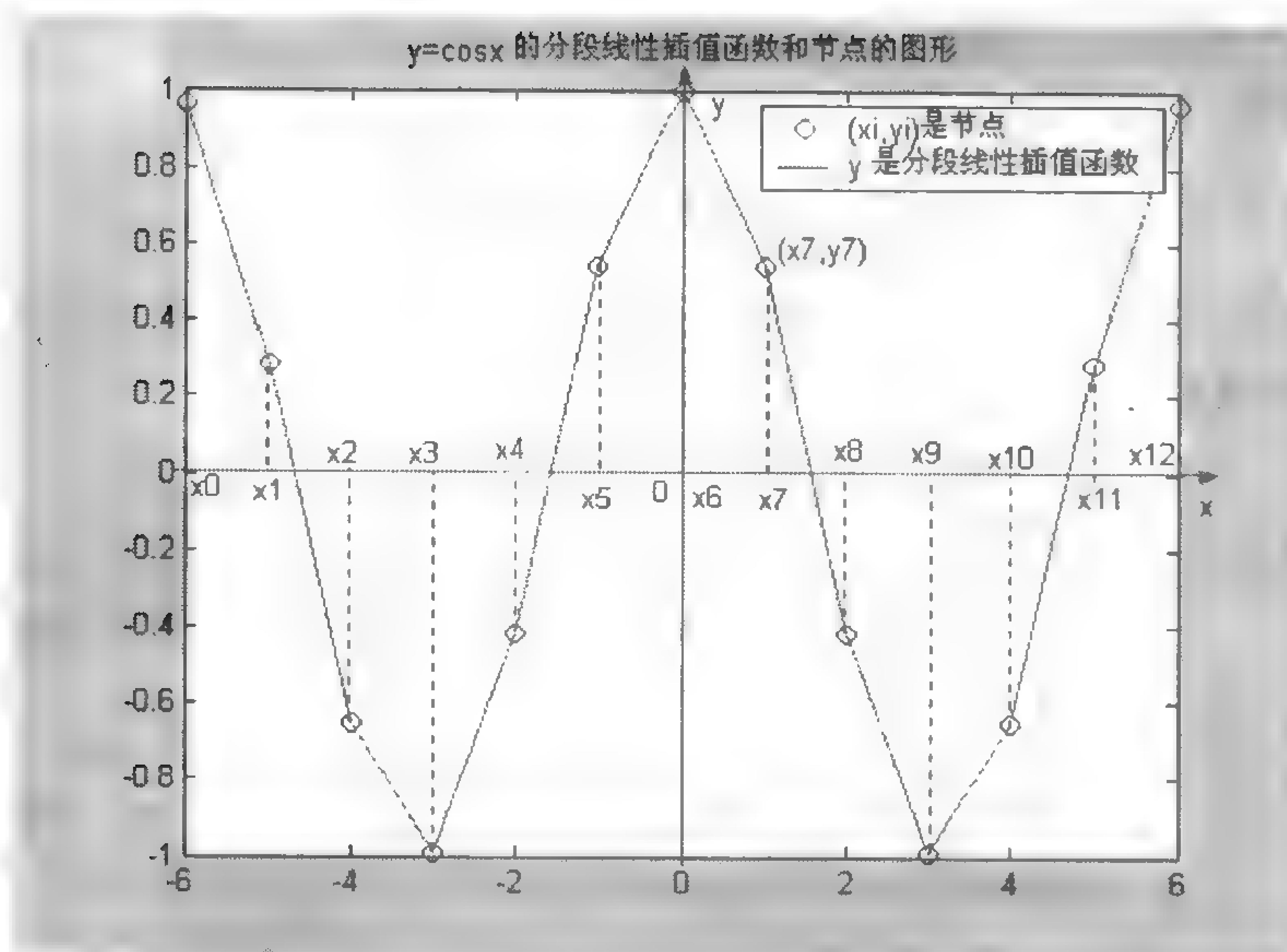


图 6－2　$y=\cos x$ 的分段线性插值函数和节点的图形

或

$$S_i(x)=l_{i-1}(x)y_{i-1}+l_i(x)y_i \quad (x_{i-1}\leqslant x\leqslant x_i, i=1,2,\cdots,n),$$

其中基函数 $l_{i-1}(x)=\dfrac{x-x_i}{x_{i-1}-x_i}, l_i(x)=\dfrac{x-x_{i-1}}{x_i-x_{i-1}}$.

(2) $S_n(x)$ 在 $[a,b]$ 上可以表示为分段函数

$$S_n(x)=\begin{cases}\dfrac{x-x_1}{x_0-x_1}y_0+\dfrac{x-x_0}{x_1-x_0}y_1, & x_0\leqslant x\leqslant x_1,\\ \cdots\cdots\cdots\cdots & \\ \dfrac{x-x_i}{x_{i-1}-x_i}y_{i-1}+\dfrac{x-x_{i-1}}{x_i-x_{i-1}}y_i, & x_{i-1}\leqslant x\leqslant x_i,\\ \cdots\cdots\cdots\cdots & \\ \dfrac{x-x_n}{x_{n-1}-x_n}y_{n-1}+\dfrac{x-x_{n-1}}{x_n-x_{n-1}}y_n, & x_{n-1}\leqslant x\leqslant x_n.\end{cases} \tag{6.46}$$

(3) $l_i(x)$ 是分段线性连续函数，且它满足

$$l_i(x_j)=\begin{cases}1, & i=j,\\ 0, & i\neq j.\end{cases} \tag{6.47}$$

(4) $S_n(x)$ 有良好的收敛性，即对于 $x\in[a,b]$ 有 $\lim\limits_{n\to\infty}S_n(x)=f(x)$.

用 $S_n(x)$ 计算插值点 x 处的插值时，只用到 x 左右的两个节点，计算量与节

点个数 n 无关. 但 n 越大,分段越多,插值误差越小. 实际上用函数表作插值计算时,分段线性插值就足够了,如数学物理中用的特殊函数表,数理统计中用的概率分布表等.

6.5.3 分段线性插值的 MATLAB 程序

分段线性插值在 MATLAB 函数库中有现成的程序,在计算时只要直接调用即可,详细的调用方法见表 6-3 和例题.

表 6-3 分段线性插值的 MATLAB 函数

命令	功能
YI = interp1(X,Y,XI)	interp1(X,Y,XI)命令的主要功能是计算函数 $\boldsymbol{Y}$ 在插值点向量 $\boldsymbol{XI}$ 的元素处的内线性插值所对应的向量 $\boldsymbol{YI}$. (1)如果输入的节点($\boldsymbol{X}(i)$,$\boldsymbol{Y}(i)$)的横坐标向量 $\boldsymbol{X}$ 是 n 维向量,则纵坐标向量 $\boldsymbol{Y}$ 也应该是 n 维. 插值点向量 $\boldsymbol{XI}$ 是 m 维,则运行后输出的插值向量 $\boldsymbol{YI}$ 也是 m 维. 参考例 6.5.2(1)和例 6.5.4 等. (2)如果输入的节点($\boldsymbol{X}(i)$,$\boldsymbol{Y}(i)$)的横坐标向量 $\boldsymbol{X}$ 是 n 维,纵坐标矩阵 $\boldsymbol{Y}_{n\times t}$ 的行数必须是 $\boldsymbol{X}$ 的维数 n,插值点向量 $\boldsymbol{XI}$ 是 m 维,则按矩阵 $\boldsymbol{Y}$ 的每列进行插值运算,运行后输出的插值矩阵 $\boldsymbol{YI}$ 是 m 行 t 列. 参考例 6.5.2(2).
YI = interp1(Y,XI)	interp1(Y,XI)命令与查表的作用相同. 这表指的是[$\boldsymbol{X}$,$\boldsymbol{Y}$],要查在 $\boldsymbol{X}$ 的元素之间位置处 $\boldsymbol{XI}$ 的元素的插值 $\boldsymbol{YI}$ 的元素,即返回值. interp1(Y,XI)命令主要用于节点($\boldsymbol{X}(i)$,$\boldsymbol{Y}(i)$)的横坐标向量 $\boldsymbol{X}$ 的元素是 1 到 n 的自然数. 由于 interp1(Y,XI)是 interp1(X,Y,XI)的特例,所以两者输入和输出的向量或矩阵的大小类似. 即 (1)如果输入的节点($\boldsymbol{X}(i)$,$\boldsymbol{Y}(i)$)的横坐标向量 $\boldsymbol{X}$ 是 n 维的,则纵坐标向量 $\boldsymbol{Y}$ 也应该是 n 维. 插值点向量 $\boldsymbol{XI}$ 是 m 维,则运行后输出的插值向量 $\boldsymbol{YI}$ 也是 m 维. 参考例 6.5.3(1). (2)如果输入的节点($\boldsymbol{X}(i)$,$\boldsymbol{Y}(i)$)的横坐标向量 $\boldsymbol{X}$ 的元素是 1 到 n 的自然数,纵坐标矩阵 $\boldsymbol{Y}_{n\times t}$ 的行数必须是 $\boldsymbol{X}$ 的维数 n,插值点向量 $\boldsymbol{XI}$ 是 m 维,则按矩阵 $\boldsymbol{Y}$ 的每列进行插值运算,运行后输出的插值矩阵 $\boldsymbol{YI}$ 是 m 行 t 列. 参考例 6.5.3(2).

例 6.5.2 给定节点($\boldsymbol{X}(i)$,$\boldsymbol{Y}(i)$)的横坐标向量 $\boldsymbol{X}$,纵坐标向量或矩阵 $\boldsymbol{Y}$,插值点向量 $\boldsymbol{XI}$ 如下,计算分段线性插值向量 $\boldsymbol{YI}$.

(1)$\boldsymbol{X}=(-5,-3,-2,5)$,$\boldsymbol{Y}=(2,3,4,5)$,$\boldsymbol{XI}=1.375$;

(2) $\boldsymbol{X}=(-5,-3,-2,5)$, $\boldsymbol{Y}=\begin{pmatrix} 2 & 3 & 4 \\ -5 & -3 & -2 \\ -5/2 & -21/10 & -19/10 \\ -1 & 2 & 5 \end{pmatrix}$, $\boldsymbol{XI}$ 是 -4 到 4 的整数.

解 (1)在 MATLAB 工作窗口输入程序

```
>>X=[-5,-3,-2,5]; Y=[2,3,4,5]; XI=1.375;
YI=interp1(X,Y,XI)
```

运行后屏幕显示

```
YI =
    4.48214285714286
```

(2)在 MATLAB 工作窗口输入程序

```
>>X=[-5, 3,-2,5]; Y=[2,3,4;-5,-3,-2;-2.5,-2.1,-1.9;-1,2,5];
XI=-4:4;YI=interp1(X,Y,XI)
```

运行后屏幕显示结果

```
YI =
  -1.50000000000000                   0    1.00000000000000
   5.00000000000000   -3.00000000000000   -2.00000000000000
  -2.50000000000000   -2.10000000000000   -1.90000000000000
  -2.28571428571429   -1.51428571428571   -0.91428571428571
  -2.07142857142857   -0.92857142857143    0.07142857142857
  -1.85714285714286   -0.34285714285714    1.05714285714286
  -1.64285714285714    0.24285714285714    2.04285714285714
  -1.42857142857143    0.82857142857143    3.02857142857143
  -1.21428571428571    1.41428571428571    4.01428571428571
```

例 6.5.3 给定节点$(\boldsymbol{X}(i),\boldsymbol{Y}(i))$的横坐标向量 $\boldsymbol{X}$ 的元素是 1 到 4 的正整数,纵坐标向量或矩阵 $\boldsymbol{Y}$,插值点向量 $\boldsymbol{XI}$ 与例 6.5.2 相同,计算分段线性插值向量 $\boldsymbol{YI}$.

解 (1)在 MATLAB 工作窗口输入程序

```
>>Y=[2,3,4,5]; XI=1.375; YI=interp1(Y,XI)
```

运行后屏幕显示结果

```
YI =
    2.37500000000000
```

(2)在 MATLAB 工作窗口输入程序

```
>>Y=[2,3,4;-5,-3,-2;-2.5,-2.1,-1.9;-1,2,5]; XI=-4:4;
YI=interp1(Y,XI)
```

运行后屏幕显示

```
YI =

        NaN            NaN            NaN
              ......................
        NaN            NaN            NaN


2.00000000000000    3.00000000000000    4.00000000000000
-5.00000000000000   -3.00000000000000   -2.00000000000000
-2.50000000000000   -2.10000000000000   -1.90000000000000
-1.00000000000000    2.00000000000000    5.00000000000000
```

例 6.5.4 设函数$f(x)=\dfrac{1}{1+25x^2}$,在区间$[-1,1]$上取等距节点(x_i,y_i),$i=0,1,2,\cdots,10$,求$f(x)$的分段线性插值函数$S_n(x)$,并用 MATLAB 计算由相邻节点构成的子区间$[x_{i-1},x_i]$的中点处$S_n(x)$的值及其相对误差.

解 (1)记节点的横坐标$x_i=-1+ih,h=0.2,i=0,1,2,\cdots,10$,插值点$x_{i-\frac{1}{2}}=\dfrac{1}{2}(x_{i-1}+x_i),i=1,2,\cdots,10$. 则

① 分段线性插值函数$S_n(x)$为

$$S_n(x)=\frac{x-x_i}{x_{i-1}-x_i}f(x_{i-1})+\frac{x-x_{i-1}}{x_1-x_{i-1}}f(x_i)\ \ (x_{i-1}\leqslant x\leqslant x_i,i=1,2,\cdots,10).$$

② $S_n(x)$在小区间$[x_{i-1},x_i],i=1,2,\cdots,10$ 的中点$x_{i-\frac{1}{2}}=\dfrac{1}{2}(x_{i-1}+x_i),i=1,2,\cdots,10$ 处的插值为

$$S_n(x_{i-\frac{1}{2}})=\frac{1}{2}[f(x_{i-1})+f(x_i)],i=1,2,\cdots,10.$$

③ 插值$S_n(x_{i-\frac{1}{2}})$的相对误差公式可以表示为

$$|R_n(x_{i-\frac{1}{2}})|=\left|\frac{f(x_{i-\frac{1}{2}})-S_n(x_{i-\frac{1}{2}})}{f(x_{i-\frac{1}{2}})}\right|.$$

(2)下面用 MATLAB 程序计算各小区间的中点处插值$S_n(x_{i-\frac{1}{2}})$及其相对误差$|R_n(x_{i-\frac{1}{2}})|$的值.

在 MATLAB 工作窗口输入程序

```
>>h=0.2;x0=-1:h:1;y0=1./(1+25.*x0.^2);
xi=-0.9:h:0.9; fi=1./(1+25.*xi.^2);
yi=interp1(x0,y0,xi); Ri=abs((fi-yi)./fi);
xi,fi,yi,Ri,i=[xi',fi',yi',Ri']
```

运行后屏幕显示小区间中点x_i,在中点x_i处的函数值f_i,插值y_i和相对误差R_i的值如下

```
xi =
   Columns 1 through 7
 -0.9000   -0.7000   -0.5000   -0.3000   -0.1000   0.1000   0.3000
   Columns 8 through 10
0.5000   0.7000   0.9000
fi =
   Columns 1 through 7
0.0471   0.0755   0.1379   0.3077   0.8000   0.8000   0.3077
   Columns 8 through 10
0.1379   0.0755   0.0471
yi =
   Columns 1 through 7
0.0486   0.0794   0.1500   0.3500   0.7500   0.7500   0.3500
   Columns 8 through 10
0.1500   0.0794   0.0486
Ri =
Columns 1 through 7
0.0337   0.0522   0.0875   0.1375   0.0625   0.0625   0.1375
  Columns 8 through 10
0.0875   0.0522   0.0337
```

6.5.4 作有关分段线性插值图形的 MATLAB 程序

如果能在同一个坐标系中作出在插值区间$[a,b]$上的节点$(x_i,f(x_i))$，被插值函数$f(x)$，$f(x)$的分段线性插值函数$S_n(x)$和插值点$(x_j,S_n(x_j))$等图形，则更有利于我们进一步直观地研究我们关心的问题. 为此编写了下面的 MATLAB 程序.

作有关分段线性插值图形的 MATLAB 主程序

输入的量：$n+1$个节点$(x_i,f(x_i))$ $(i=1,2,\cdots,n+1)$横坐标向量x_0，纵坐标向量y_0，插值点$(x_j,S_n(x_j))$横坐标向量x和函数$y=f(x)$值.

输出的量：插值$s=S_n(x_j)$.

输出的图形：在插值区间$[a,b]$上的节点$(x_i,f(x_i))$，被插值函数$f(x)$，$f(x)$的分段线性插值函数$S_n(x)$和插值点$(x_j,S_n(x_j))$.

```
function s = xxczhjt1(x0,y0,xi,x,y)
s = interp1(x0,y0,xi);
Sn = interp1(x0,y0,x0);
plot(x0,y0,'o',x0,Sn,'-',xi,s,'*',x,y,'-.')
```

```
legend('节点(xi,yi)','分段线性插值函数 Sn(x)','插值点(x,s)','被插值函
数 y')
```

我们也可以直接在 MATLAB 工作窗口编程序. 例如,

```
>>x0 = -6:6; y0 =sin(x0);
xi = -6:.25:6;
yi =interp1(x0,y0,xi);
x = -6:0.001:6; y =sin(x);plot(x0,y0,'o',xi,yi,x,y,':'),
legend('节点(xi,yi)','分段线性插值函数','y =sinx 的函数')
>>x0 - -6:6; y0 -cos(x0);
xi - -6:.25:6;
yi =interp1(x0,y0,xi);
x = -6:0.001:6; y =cos(x); plot(x0,y0,'o',xi,yi,x,y,':'),
legend('节点(xi,yi)','分段线性插值函数','y =cosx 的函数')
```

例 6.5.5 设函数 $f(x)=\dfrac{1}{1+25x^2}$,在区间[-1,1]上取等距节点 (x_i,y_i),$i=0,1,2,\cdots,10$,构造分段线性插值函数 $S_n(x)$,用 MATLAB 程序计算各小区间的中点 x_i 处 $S_n(x)$ 的值,作出节点,插值点,$f(x)$ 和 $S_n(x)$ 的图形.

解 节点的横坐标和插值点等取值与例 6.5.4 相同. 在 MATLAB 工作窗口输入程序

```
>>x0 = -1:0.2:1; y0 =1./(1 +25.*x0.^2);
xi = -0.9:0.2:0.9;
b =max(x0);
a =min(x0);x =a:0.001:b;
y =1./(1 +25.*x.^2);
s =xxczhjt1(x0,y0,xi,x,y),
title('y =1/(1 +25 x^2)的分段线性插值的有关图形')
```

运行后屏幕显示各小区间中点 x_i 处 $S_n(x)$ 的值,出现节点、插值点、$f(x)$ 和 $S_n(x)$ 的图形(见图 6 -3).

```
s =
    Columns 1 through 4
    0.04864253393665  0.07941176470588  0.15000000000000  0.35000000000000
    Columns 5 through 8
    0.75000000000000  0.75000000000000  0.35000000000000  0.15000000000000
    Columns 9 through 10
    0.07941176470588  0.04864253393665
```

例 6.5.6 设函数 $f(x)=0.5x-\sin x$,在区间[$-\pi,\pi$]上取等距节点 (x_i,y_i),$i=0,1,2,\cdots,7$,构造分段线性插值函数 $S_n(x)$,用 MATLAB 程序计算各小

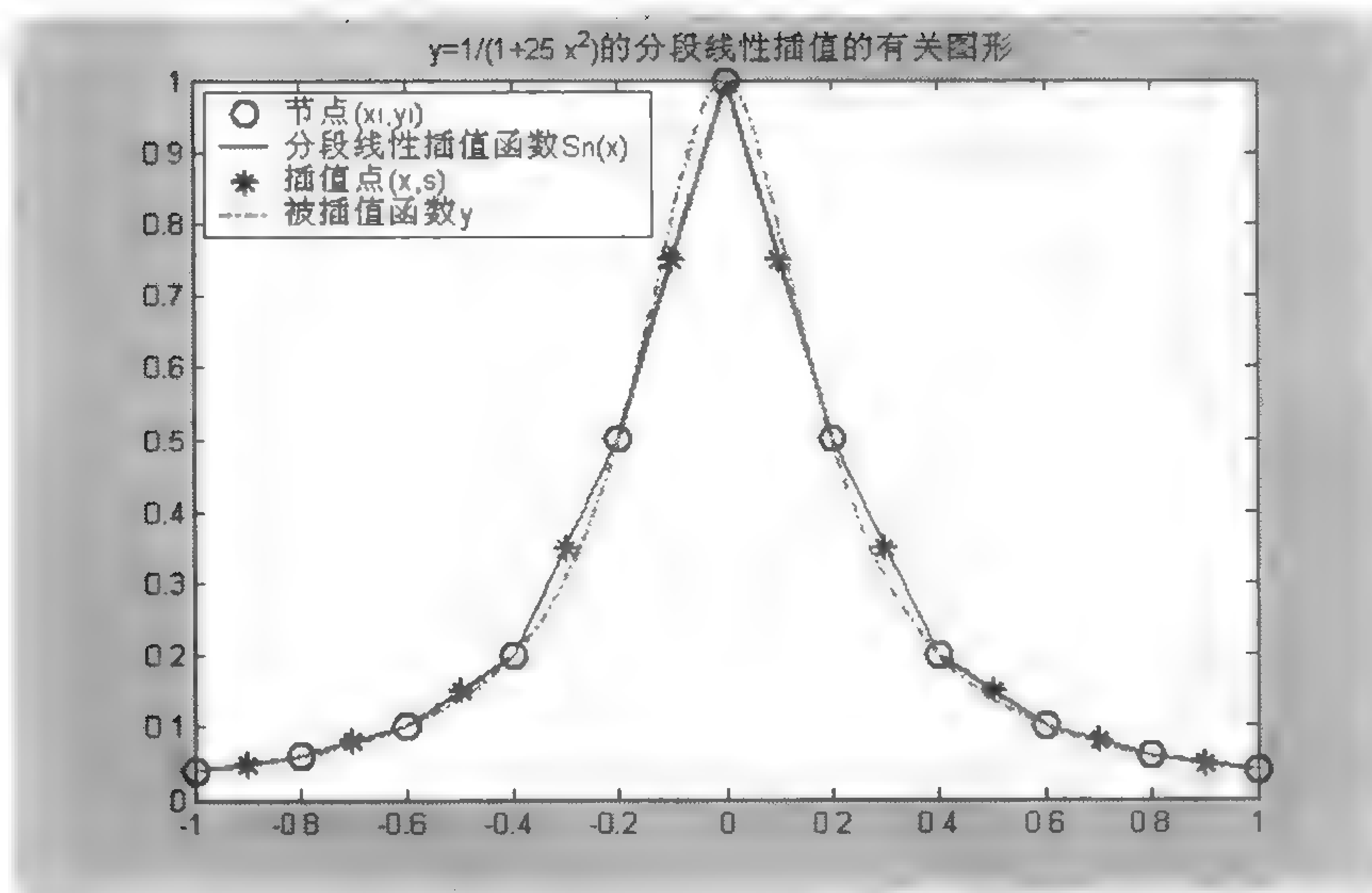


图 6-3 $f(x)=\dfrac{1}{1+25x^2}$的分段线性插值的有关图形

区间中点 x_i 处 $S_n(x)$的值,作出节点,插值点,$f(x)$和 $S_n(x)$的图形.

解 记节点的横坐标 $x_i=-\pi+ih,h=2\pi/7,i=0,1,2,\cdots,7$,插值点 $x_{i+\frac{1}{2}}=\frac{1}{2}(x_i+x_{i+1}),i=0,1,2,\cdots,6$. 在 MATLAB 工作窗口输入程序

```
>> h=2*pi/7; x0 = -pi:h:pi; y0 =0.5.*x0 - sin(x0);
xi = -pi+h/2:h:pi  h/2; b=max(x0);
a=min(x0);x=a:0.001:b; y=0.5.*x-sin(x);
s=xxczhjt1(x0,y0,xi,x,y)
title('y=0.5x-sin(x)及其分段线性插值函数,插值,节点的图形')
```

运行后屏幕显示各小区间中点 x_i 处 $S_n(x)$的值,出现节点,插值点,$f(x)$和$S_n(x)$的图形(见图 6-4)如下

```
s =
   Columns 1 through 4
    -0.9555   -0.01922   0.2556   0
   Columns 5 through 7
    -0.2556   0.0192   0.9555
s = -0.9555 -0.0192  0.2556  0  -0.2556  0.0192  0.9555
```

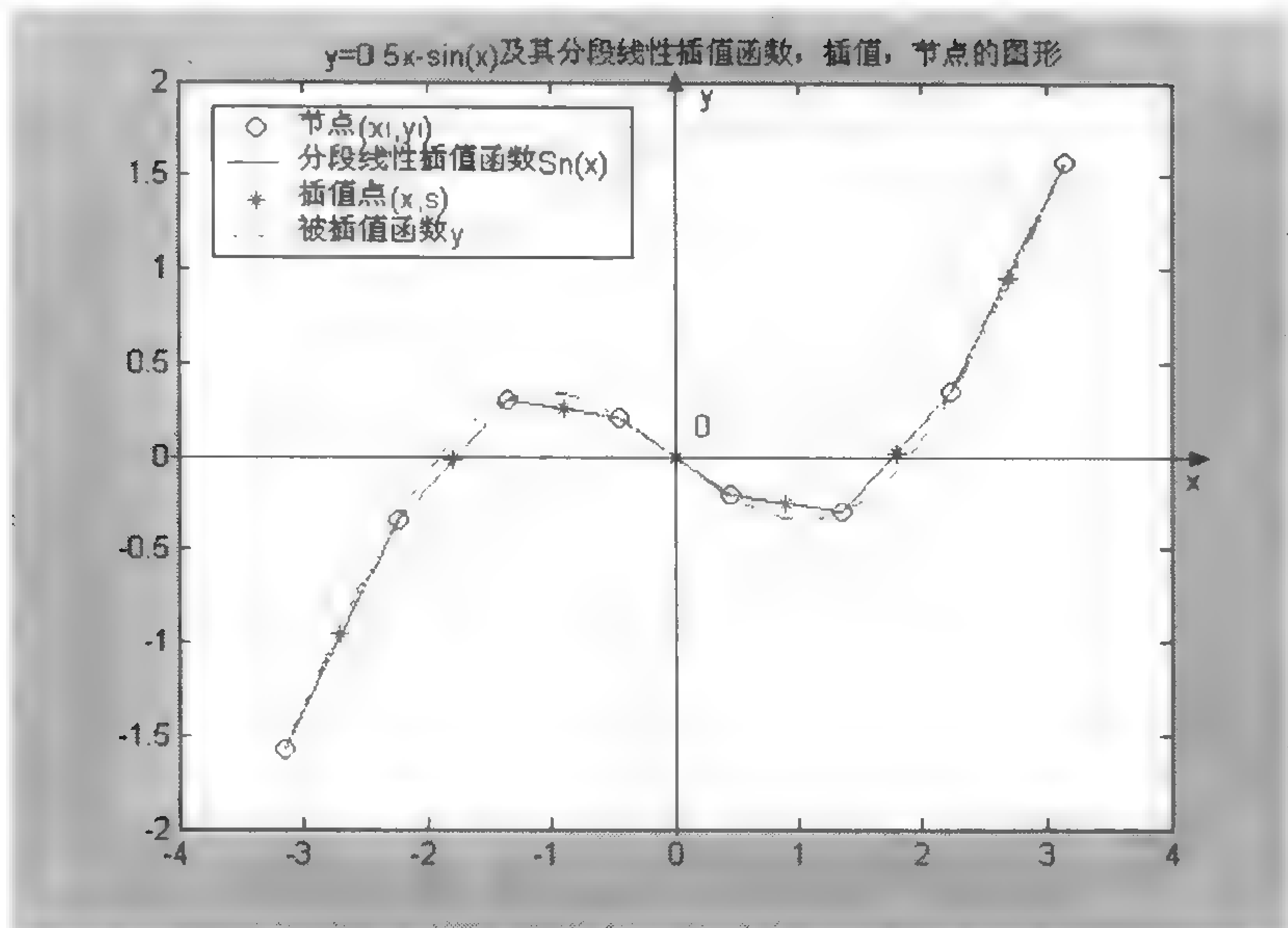


图 6-4　$y=0.5x-\sin(x)$及其分段线性插值函数,插值,节点的图形

6.5.5　用 MATLAB 计算有关分段线性插值的误差

定理 6.8　如果被插值函数$f(x)$在插值区间$[a,b]$上具有二阶连续导数,$[a,b]$上的$n+1$个节点(x_i,y_i) $(i=0,1,2,\cdots,n)$满足$a=x_0<x_1<x_2<\cdots<x_n=b$,$S_n(x)$是定义 6.3 所定义的$f(x)$在$[a,b]$上的分段线性插值函数,则

(1) 被插值函数$f(x)$在由相邻节点(x_i,y_i) $(i=0,1,2,\cdots,n)$构成的子区间$[x_{i-1},x_i]$上的$S_n(x)$的误差公式可以表示为

$$|R_n(x)|=|f(x)-S_n(x)|\leqslant\frac{(x_i-x_{i-1})^2}{8}\max_{x_{i-1}\leqslant x\leqslant x_i}|f''(x)|. \tag{6.48}$$

(2) $S_n(x)$在插值区间$[a,b]$上的误差公式可以表示为

$$|R_n(x)|=|f(x)-S_n(x)|\leqslant\frac{\max\limits_{1\leqslant i\leqslant n}|x_i-x_{i-1}|^2}{8}\max_{a\leqslant x\leqslant b}|f''(x)|. \tag{6.49}$$

且$S_n(x)$在$[a,b]$上一致收敛到$f(x)$.

因为等距节点所构造的子区间$[x_{i-1},x_i]$的长度相等,所以$\max\limits_{1\leqslant i\leqslant n}|x_i-x_{i-1}|^2=|x_i-x_{i-1}|^2=h^2$. 由定理 6.8 可以得到下面的推论.

推论 6.1　如果被插值函数$f(x)$在插值区间$[a,b]$上具有二阶连续导数,$[a,b]$上的$n+1$个节点(x_i,y_i) $(i=0,1,2,\cdots,n)$的横坐标是等距离的点,即

$x_i=a+h\cdot i$,其中 $h=\dfrac{b-a}{n}$ $(i=0,1,2,\cdots,n)$,$S_n(x)$是定义 6.3 所定义的$f(x)$在$[a,b]$上的分段线性插值函数,则

(1)被插值函数$f(x)$在由相邻节点(x_i,y_i) $(i=0,1,2,\cdots,n)$构成的子区间$[x_{i-1},x_i]$上的 $S_n(x)$的误差公式可以表示为

$$|R_n(x)|=|f(x)-S_n(x)|\leqslant\frac{h^2}{8}\max_{x_{i-1}\leqslant x\leqslant x_i}|f''(x)|. \tag{6.50}$$

(2) $S_n(x)$在插值区间$[a,b]$上的误差公式可以表示为

$$|R_n(x)|=|f(x)-S_n(x)|\leqslant\frac{h^2}{8}\max_{a\leqslant x\leqslant b}|f''(x)|. \tag{6.51}$$

例 6.5.7 设函数$f(x)=\dfrac{1}{1+25x^2}$,在区间$[-1,1]$上取等距节点(x_i,y_i),$i=0,1,2,\cdots,10$,构造分段线性插值函数 $S_n(x)$,用 MATLAB 程序计算 $S_n(x)$在$[-1,1]$上的误差公式.

解 (1) 节点的横坐标和插值点等取值与例 6.5.4 相同. 则 $S_n(x)$在区间$[-1,1]$上的误差公式可以表示为

$$|R_n(x)|=|f(x)-S_n(x)|\leqslant\frac{h^2}{8}\max_{-1\leqslant x\leqslant 1,}|f''(x)|,$$

其中 $h=0.2$.

(2)在 MATLAB 工作窗口输入程序如下

```
>> syms x y,y =1 /(1 +25 * x^2); yxx =diff(y,x,2),
```

运行后输出$f(x)$的二阶导数为

```
yxx =
      5 000 /(1 +25 * x^2)^3 * x^2 -50 /(1 +25 * x^2)^2
```

(3) 在 MATLAB 工作窗口输入程序如下

```
>> syms h,x = -1:0.0001:1;
yxx =5000./(1 +25.* x.^2).^3.* x.^2 -50./(1 +25.* x.^2).^2;
myxx =max(yxx), R =(h^2)* myxx/8
```

运行后输出$f(x)$的二阶导数在区间$[-1,1]$上的最大值 *myxx* 和 $S_n(x)$在区间$[-1,1]$上的误差公式 R 为

```
myxx =              R =
      12.500 0          25/16 * h^2
```

例 6.5.8 设函数$f(x)=\tan\left(\cos\left(\dfrac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$,在区间$[-\pi,\pi]$上取等距节点$(x_i,y_i)$,$i=0,1,2,\cdots,9$,构造分段线性插值函数 $S_n(x)$.

(1) 用 MATLAB 程序计算各小区间中点 x_i 处 $S_n(x)$的值,作出节点、插值

点 $f(x)$ 和 $S_n(x)$ 的图形；

（2）用 MATLAB 程序计算各小区间中点处 $S_n(x)$ 的值及其相对误差；

（3）用 MATLAB 程序估计 $\max\limits_{-\pi\leq x\leq\pi}|f''(x)|$ 和 $S_n(x)$ 在区间 $[-\pi,\pi]$ 上的误差限.

解 （1）记节点的横坐标 $x_i=-\pi+ih, h=2\pi/9, i=0,1,2,\cdots,9$，插值点 $x_{i+\frac{1}{2}}=\frac{1}{2}(x_i+x_{i+1})$. $i=0,1,2,\cdots,8$. $R_i=\left|\frac{f(x_{i+\frac{1}{2}})-S_n(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right|$.

在 MATLAB 工作窗口输入程序

```
>>h=2*pi/9; x0=-pi:h:pi;
y0-tan(cos((3^(1/2)+sin(2*x0))./(3+4*x0.^2)));
xi=-pi+h/2:h:pi-h/2;
fi=tan(cos((3^(1/2)+sin(2*xi))./(3+4*xi.^2)));
b=max(x0); a=min(x0);
x=a:0.001:b; y=tan(cos((3^(1/2)+sin(2.*x))./(3+4*x.^2)));
si=xxczhjt1(x0,y0,xi,x,y);
Ri=abs((fi-yi)./fi);
xi,fi,si,Ri,i=[xi',fi',si',Ri']
title('y=tan(cos((sqrt(3)+sin(2 x))/(3+4 x^2)))的分段线性插值
的有关图形')
```

运行后屏幕显示 R_i（略），并且作出节点、插值点 $f(x)$ 和 $S_n(x)$ 的图形（见图6-5）.

将各小区间中点 $x_i=x_{i+\frac{1}{2}}$ 处的函数值 $f_i=f_i(x_{i+\frac{1}{2}})$，插值 $s_i=S_n(x_{i+\frac{1}{2}})$，相对误差值 R_i 列表 6-4 如下.

表 6-4 例 6.5.8 的各小区间中点 x_i 处的函数值 f_i，插值 s_i 和相对误差值 R_i

i	$x_{i+\frac{1}{2}}=\frac{1}{2}(x_i+x_{i+1})$	$f(x_{i+\frac{1}{2}})$	$S_n(x_{i+\frac{1}{2}})$	$\left\|\frac{f(x_{i+\frac{1}{2}})-S_n(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right\|$
0	-2.792 526 803 190 93	1.549 179 746 366 19	1.547 314 872 437 70	0.001 203 781 506 23
1	-2.094 395 102 393 20	1.530 392 391 872 18	1.533 011 882 260 06	0.001 711 646 243 00
2	-1.396 263 401 595 46	1.529 424 914 525 70	1.530 048 627 687 94	0.000 407 808 945 91
3	-0.698 131 700 797 73	1.519 120 343 743 33	1.468 654 293 319 41	0.033 220 574 414 51
4	0	1.110 956 050 058 38	1.193 213 735 979 07	0.074 042 250 290 97
5	0.698 131 700 797 73	1.145 460 398 956 37	1.174 491 048 274 59	0.025 344 088 145 41
6	1.396 263 401 595 46	1.496 160 173 406 94	1.454 441 034 374 74	0.027 884 139 528 46
7	2.094 395 102 393 20	1.554 369 358 742 26	1.549 621 547 496 13	0.003 054 493 592 16
8	2.792 526 803 190 93	1.555 671 074 430 44	1.555 324 687 364 58	0.000 222 660 864 22

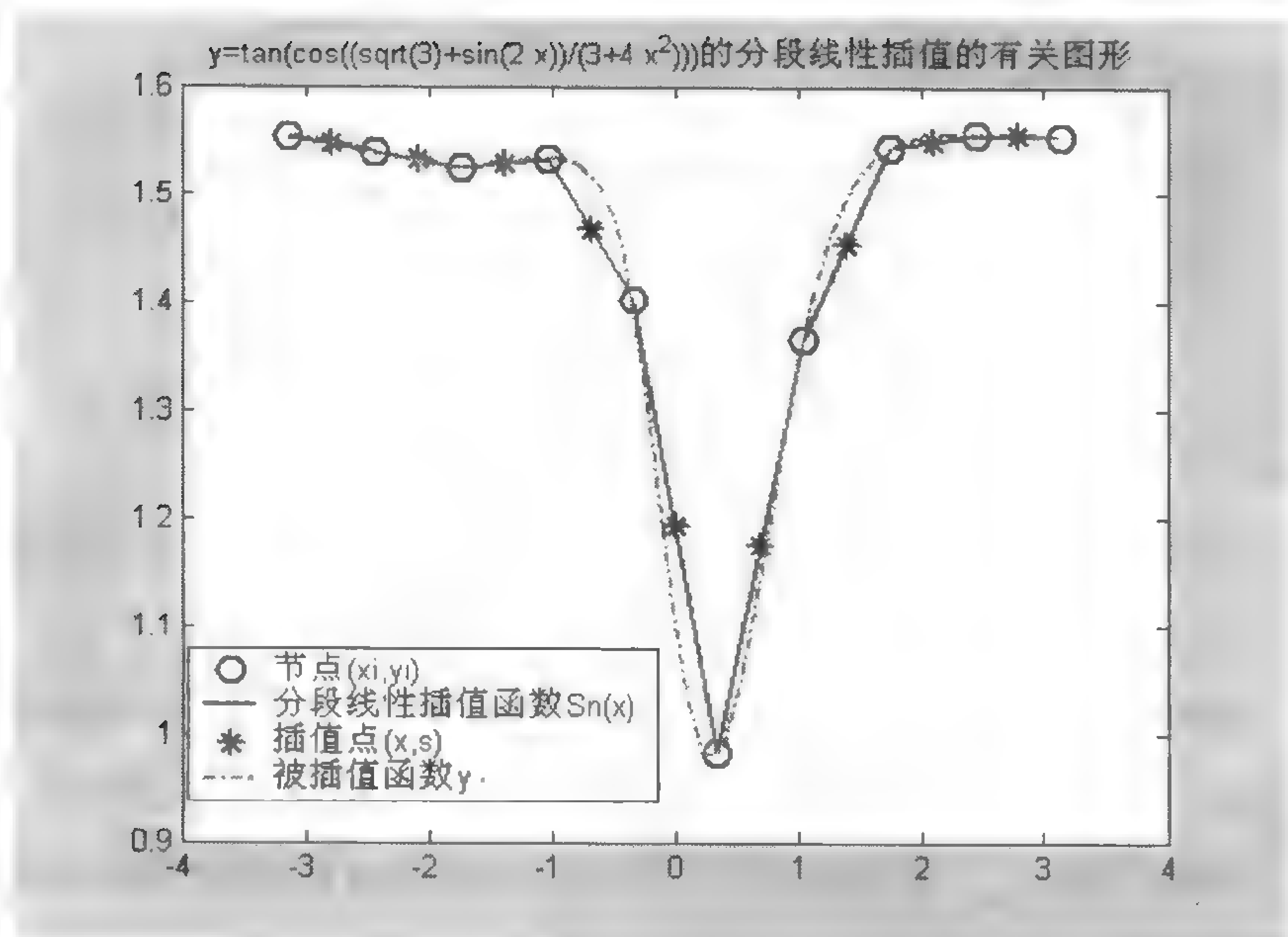


图 6-5 函数 $f(x)=\tan\left(\cos\left(\dfrac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$ 的分段线性插值的有关图形

(2)在 MATLAB 工作窗口输入程序

```
>> syms x
y=tan(cos((3^(1/2)+sin(2*x))/(3+4*x^2))); yxx=diff(y,x,2),
```

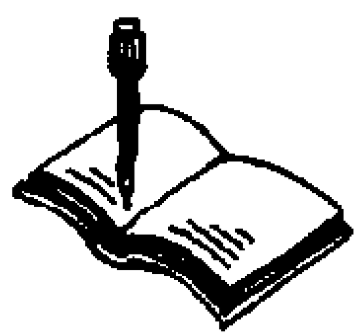
运行后屏幕显示函数 $f(x)$ 的二阶导数 $f''(x)$ (略).

在 MATLAB 工作窗口输入程序

```
>> h=2*pi/9; x=-pi:0.0001:-pi;
yxx=2*tan(cos((3^(1/2)+sin(2.*x))./(3+4.*x.^2))).*(1+tan
(cos((3.^(1/2)+sin(2.*x))./(3+4.*x.^2))).^2).*sin((3^(1/2)+sin(2.*
x))./(3+4.*x.^2)).^2.*(2*cos(2.*x)./(3+4*x.^2)-8*(3^(1/2)+sin(2.
*x))./(3+4.*x.^2).^2.*x).^2-(1+tan(cos((3^(1/2)+sin(2.*x))./(3+4.
*x.^2))).^2).*cos((3^(1/2)+sin(2.*x))./(3+4.*x.^2)).*(2.*cos(2.*
x)./(3+4.*x.^2)-8*(3^(1/2)+sin(2.*x))./(3+4.*x.^2).^2.*x).^2-(1
+tan(cos((3^(1/2)+sin(2.*x))./(3+4.*x.^2))).^2).*sin((3.^(1/2)+
sin(2.*x))./(3+4.*x.^2)).*(-4.*sin(2.*x)./(3+4.*x.^2)-32*cos(2.
*x)./(3+4.*x.^2).^2.*x+128*(3^(1/2)+sin(2.*x))./(3+4.*x.^2).^3.
*x.^2-8*(3^(1/2)+sin(2.*x))./(3+4.*x.^2).^2);
myxx=max(yxx), R=(h^2)*myxx/8
```

运行后屏幕显示 $myxx = \max\limits_{-\pi\leqslant x\leqslant\pi} |f''(x)|$ 和 $S_n(x)$ 在区间 $[-\pi,\pi]$ 上的误差限 $R = \frac{h^2}{8}\max\limits_{-\pi\leqslant x\leqslant\pi} |f''(x)|$ 如下

```
myxx =                                    R =
        -0.02788637150664                        -0.00169893490711
```



习 题 6.5

1. 作函数 $f(x)=\frac{1}{1+25x^2}$ 在区间 $[-5,5]$ 上的 n 次拉格朗日插值多项式 $L_n(x)$ 的图形 $(n=2,4,6,8,10)$，并讨论插值多项式 $L_n(x)$ 的次数与误差 $|R_n(x)|$ 的关系.

2. 设函数 $f(x)=\sin\left(\frac{3-\cos 4x}{1+25x^2}\right)$ 定义在区间 $[-\pi,\pi]$ 上，取 $n=13$，按等距节点求分段线性插值函数 $S_n(x)$，并用MATLAB程序计算各小区间中点处 $S_n(x)$ 的值及其相对误差.

3. 设函数 $f(x)=\frac{1}{1+x^2}$ 定义在区间 $[-5,5]$ 上，取 $n=10$，按等距节点构造分段线性插值函数 $S_n(x)$，用 MATLAB 程序计算各小区间中点 x_i 处 $S_n(x)$ 的值，作出节点、插值点、$f(x)$ 和 $S_n(x)$ 的图形.

4. 设函数 $f(x)=0.15x^2-\sin(2x-1)$ 定义在区间 $[-\pi,\pi]$ 上，取 $n=7$，按等距节点构造分段线性插值函数 $S_n(x)$，用 MATLAB 程序计算各小区间中点 x_i 处 $S_n(x)$ 的值，作出节点、插值点、$f(x)$ 和 $S_n(x)$ 的图形.

6.6 分段埃尔米特插值及其 MATLAB 程序

虽然分段线性插值函数 $S_n(x)$ 具有良好的收敛性，但是由图 6-2 至图 6-5 可以看出，因为分段线性插值函数 $S_n(x)$ 是将每两个相邻的节点用直线段连起来形成的一条折线，所以在节点处失去了光滑性. 如果不仅要求插值函数 $S_n(x)$ 在节点处的函数值 $S_n(x_i)=f(x_i)$，而且要求 $S_n'(x_i)=f'(x_i)$，即 $S_n(x)$ 在节点处光滑，则可以采用分段埃尔米特插值. 本节介绍分段埃尔米特插值及其 MATLAB 程序.

6.6.1 分段埃尔米特插值函数

分段埃尔米特插值也属于分段插值方法. 我们可以用定义分段线性插值函数的方法，类似地给出分段埃尔米特插值函数的定义.

定义 6.4 设函数 $f(x)$ 在 $[a,b]$ 上的 $n+1$ 个点 $a=x_0<x_1<x_2<\cdots<x_n=b$

处的函数值 $f(x_i)=y_i$，一阶导数值 $f'(x_i)=y_i'$ $(i=0,1,2,\cdots,n)$. 连接每两个相邻的点 (x_i,y_i) 和 (x_{i+1},y_{i+1})，作一条曲线函数 $H_{n,3}(x)$，使得 $H_{n,3}(x)$ 满足如下条件：

(1) $H_{n,3}(x)$ 在 $[a,b]$ 上有连续的一阶导数；

(2) $H_{n,3}(x_i)=y_i, H_{n,3}'(x_i)=y_i'$ $(i=0,1,2,\cdots,n)$；

(3) $H_{n,3}(x)$ 在每个子区间 $[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$ 上是三次多项式，

则称折曲线函数 $H_{n,3}(x)$ 为**分段三次埃尔米特插值函数**.

由 6.4 节对两节点 x_0,x_1 的埃尔米特插值的讨论，用于子区间 $[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$ 上，不难构造出插值基函数，再由定义 6.4 得到分段埃尔米特插值函数.

定理 6.9 如果 $H_{n,3}(x)$ 是定义 6.4 所定义的分段埃尔米特插值函数，且 $H_{n,3}(x)$ 在每个子区间 $[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$ 上的**三阶埃尔米特插值多项式**为

$$H_i(x)=\left[\left(1+2\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)y_{i-1}+(x-x_{i-1})y_{i-1}'\right]\left(\frac{x-x_i}{x_{i-1}-x_i}\right)^2$$
$$+\left[\left(1+2\frac{x-x_i}{x_{i-1}-x_i}\right)y_i+(x-x_i)y_i'\right]\left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2,$$

或

$$H_i(x)=g_{i-1}(x)y_{i-1}+h_{i-1}(x)y_{i-1}'+g_i(x)y_i+h_i(x)y_i',$$
$$x\in[x_{i-1},x_i]\ (i=1,2,\cdots,n),$$

其中基函数

$$g_{i-1}(x)=\left(1+2\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)\cdot\left(\frac{x-x_i}{x_{i-1}-x_i}\right)^2,\ h_{i-1}(x)=(x-x_{i-1})\cdot\left(\frac{x-x_i}{x_{i-1}-x_i}\right)^2,$$

$$g_i(x)=\left(1+2\frac{x-x_i}{x_{i-1}-x_i}\right)\cdot\left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2,\ h_i(x)=(x-x_i)\cdot\left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2.$$

则分段三次埃尔米特插值函数为

$$H_{n,3}(x)=\begin{cases}H_1(x),\ x_0\leqslant x\leqslant x_1,\\ \cdots\cdots\cdots\cdots\\ H_i(x),\ x_{i-1}\leqslant x\leqslant x_i,\\ \cdots\cdots\cdots\cdots\\ H_n(x),\ x_{n-1}\leqslant x\leqslant x_n,\end{cases}\tag{6.52}$$

或

$H_{n,3}(x)=\{H_i(x)\mid H_i(x)=g_{i-1}(x)y_{i-1}+h_{i-1}(x)y'_{i-1}+g_i(x)y_i+h_i(x)y'_i,x\in[x_{i-1},x_i],i=1,2,\cdots,n\}.$

$H_{n,3}(x)$的误差比分段线性插值的误差小,且有良好的收敛性,即对于任意$x\in[a,b]$有

$$\lim_{n\to\infty}H_{n,3}(x)=f(x).$$

用$H_{n,3}(x)$计算x点的插值时,计算量与节点个数n无关.但n越大,分段越多,插值误差越小,并$H_{n,3}(x)$在节点处光滑,且与被插值函数$f(x)$的误差小.例如,图6-6和图6-7分别是函数$y=\sin x$在区间$[-6,6]$上节点(x_i,y_i)(其中$x_i=-6+1.5i(i=0,1,2,\cdots,12)$)处的分段线性插值函数和节点的图形与分段三次埃尔米特插值函数和节点的图形.

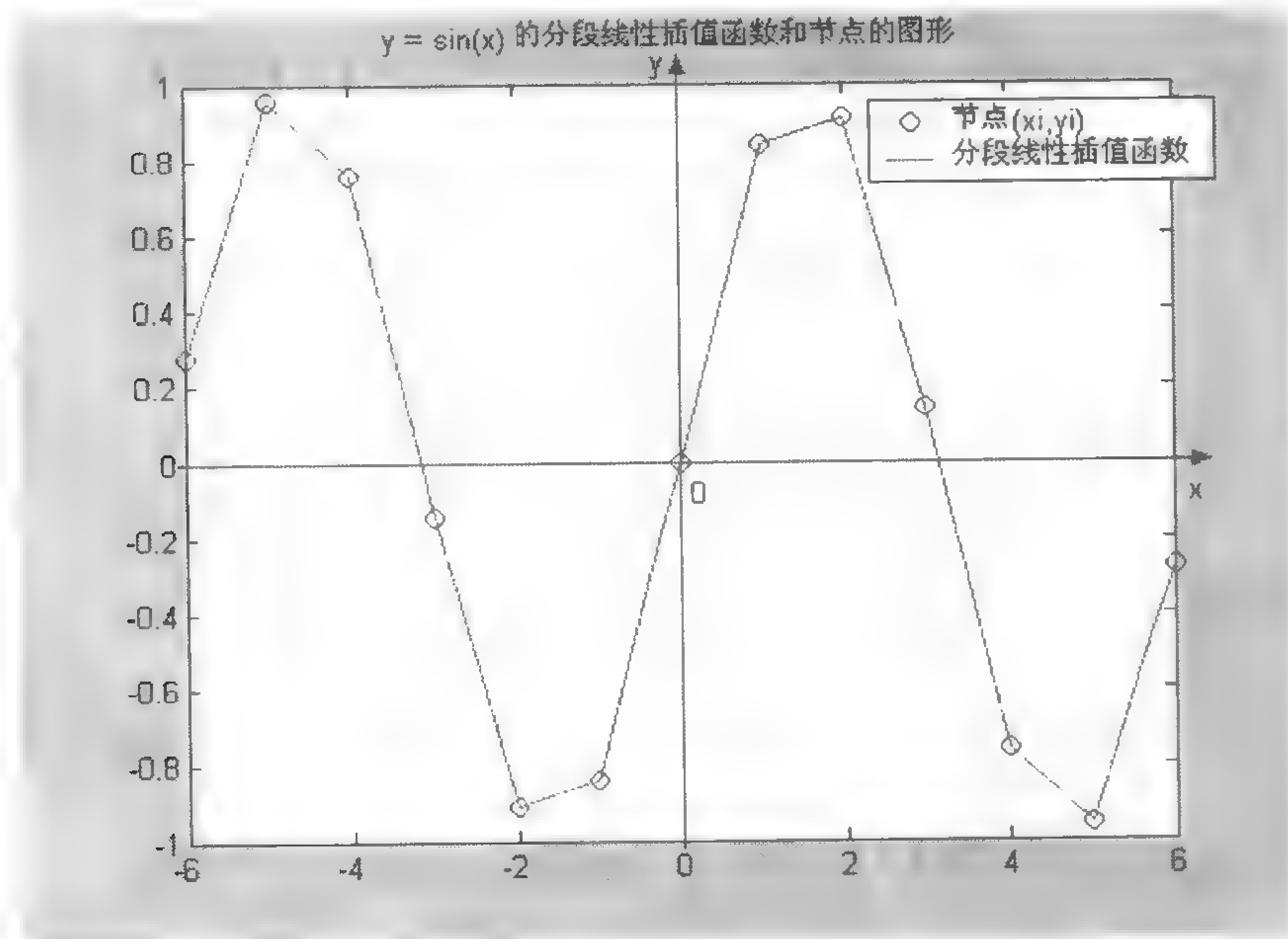


图6-6 $y=\sin x$分段线性插值函数和节点的图形

图6-8和图6-9分别是函数$y=\cos x$在区间$[-6,6]$上节点(x_i,y_i)(其中$x_i=-6+1.5i$ $(i=0,1,2,\cdots,12)$)处的分段线性插值函数和节点的图形与分段三次埃尔米特插值函数和节点的图形.小圆圈表示节点(x_i,y_i),实线表示分段插值函数.在节点处分段线性插值函数的曲线都是尖点,即不光滑.而分段

三次埃尔米特插值函数的曲线在节点处十分光滑.

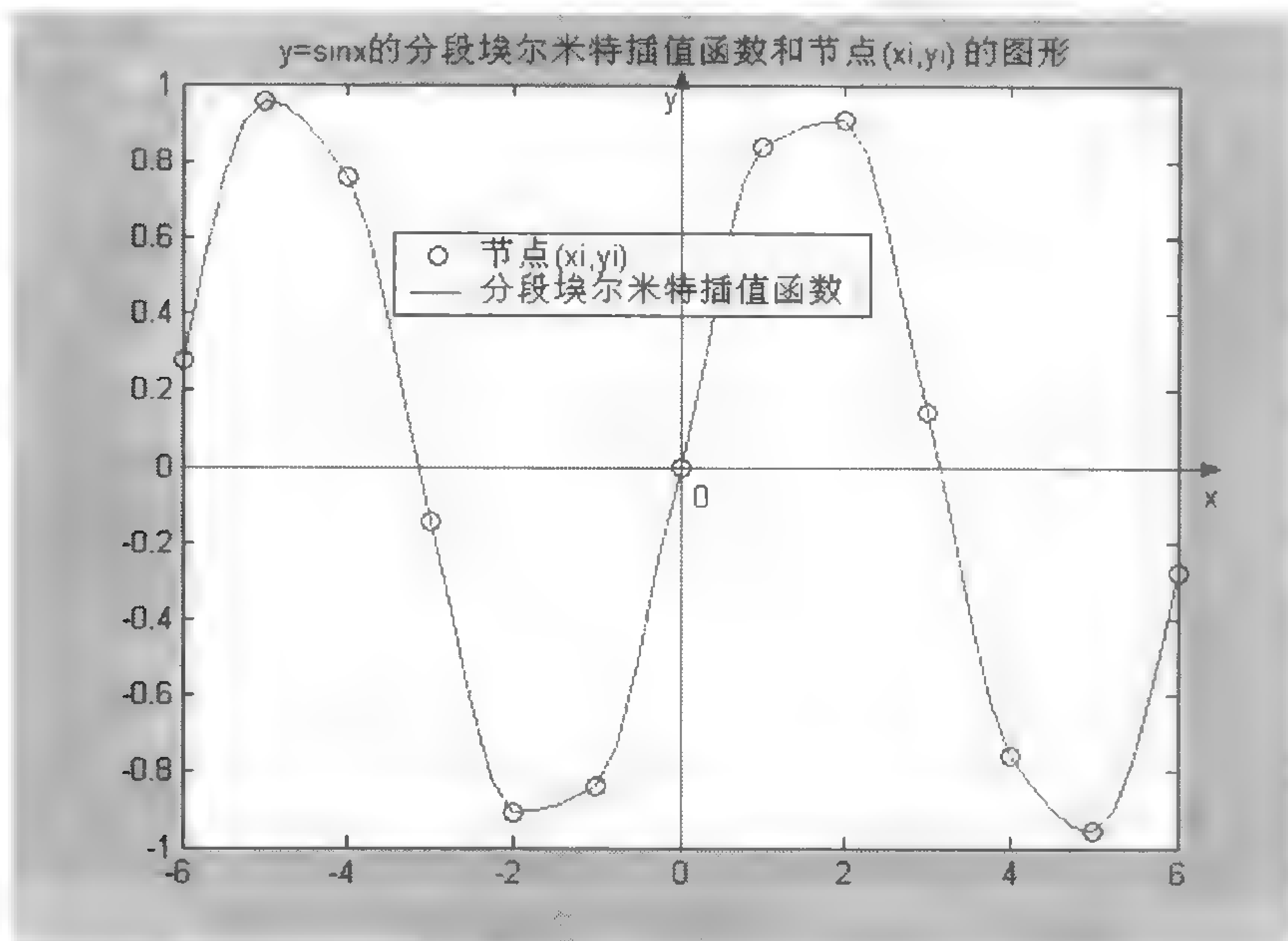


图 6-7 $y=\sin x$ 分段三次埃尔米特插值函数和节点的图形

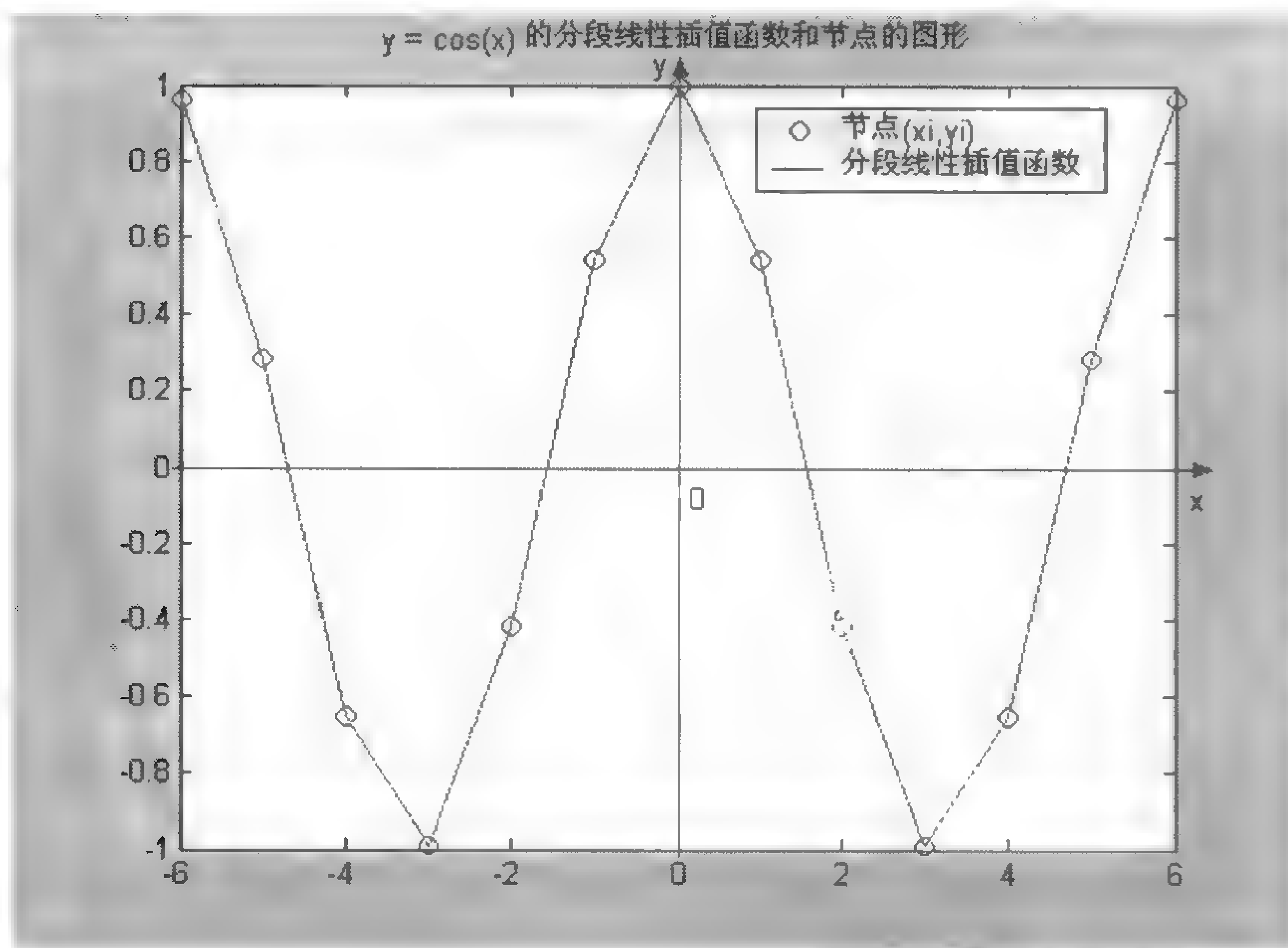


图 6-8 $y=\cos x$ 分段线性插值函数和节点的图形

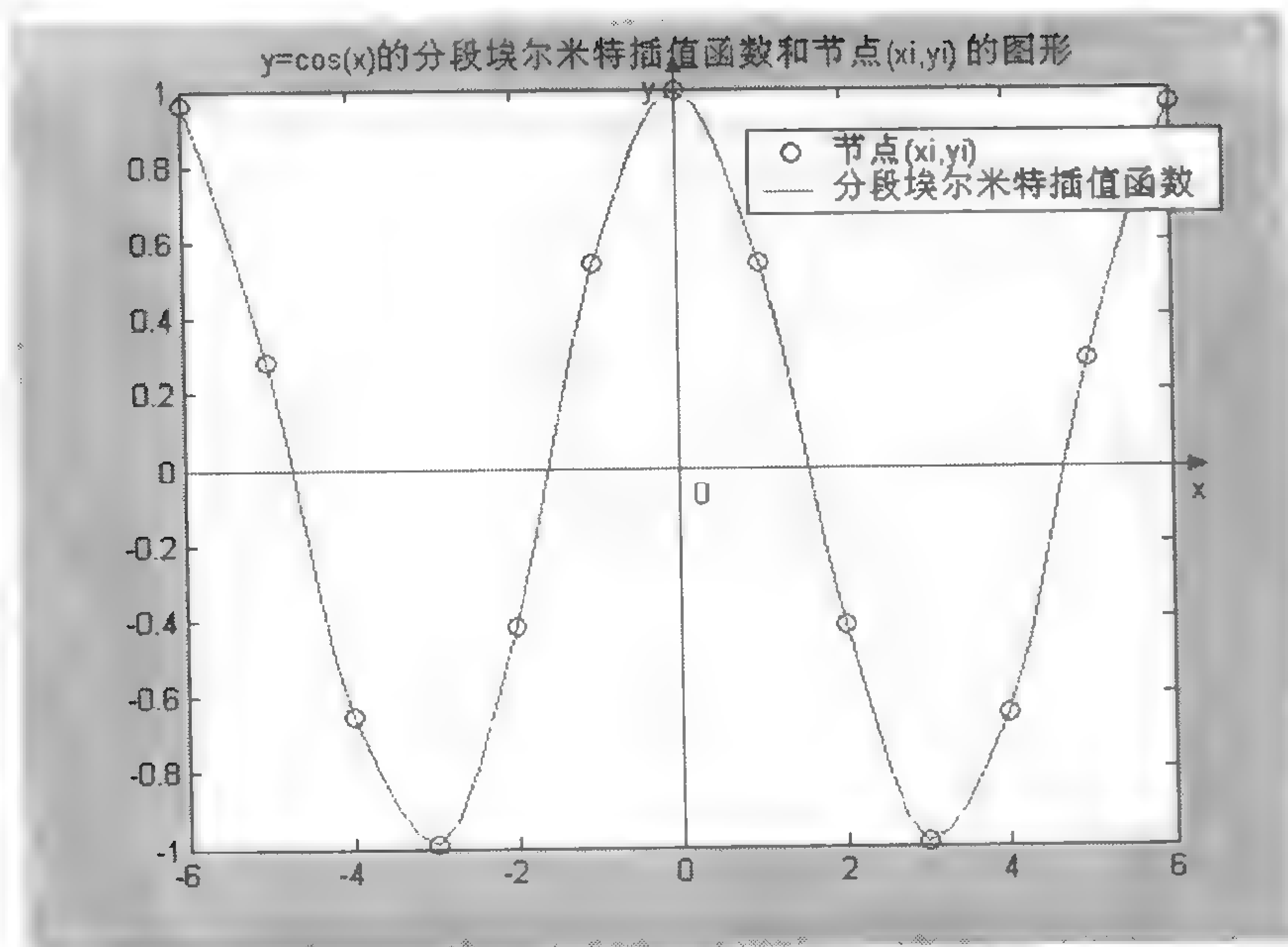


图 6-9　$y=\cos x$ 分段埃尔米特插值函数和节点的图形

再如,图 6-10 和图 6-11 分别是在区间[-6,6]上被插值函数 $y=\sin x$ 和在其节点(x_i,y_i)(其中 $x_i=-6+1.5i$ $(i=0,1,2,\cdots,12)$)处的分段线性插值函

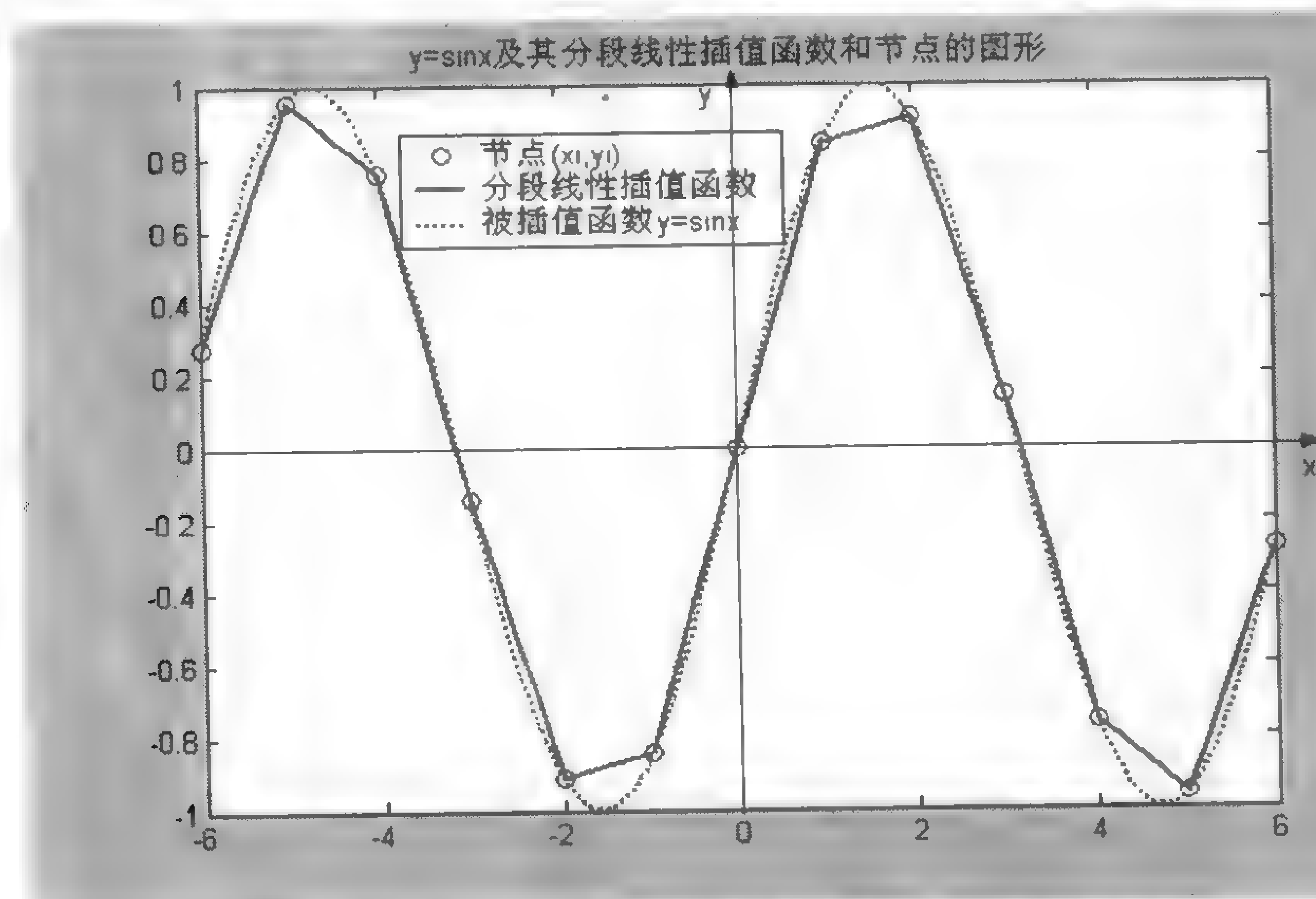


图 6-10　$y=\sin x$ 及其分段线性插值函数和节点的图形

数和节点的图形与分段三次埃尔米特插值函数和节点的图形. 图 6－12 和图

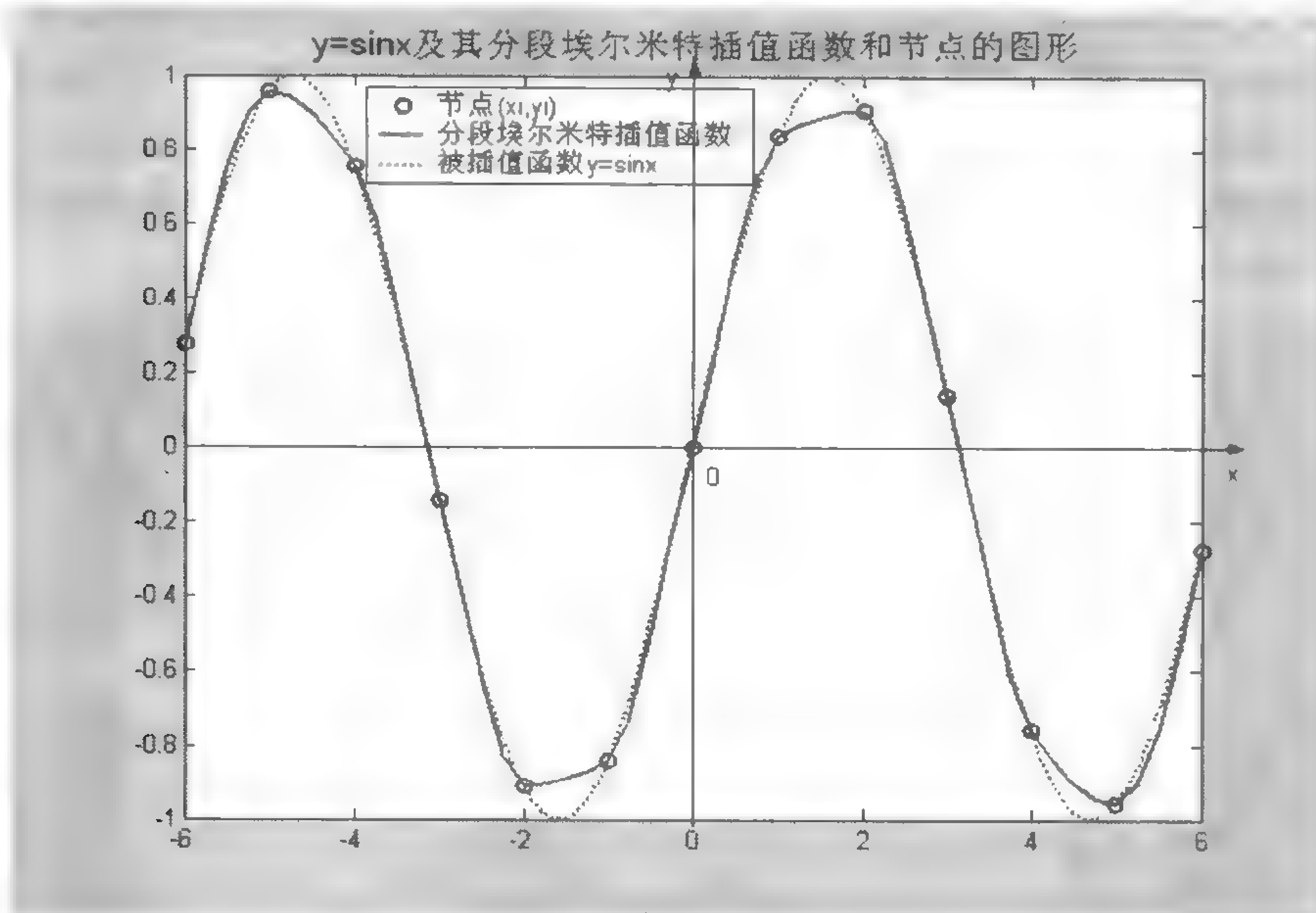


图 6－11 $y=\sin x$ 及其分段埃尔米特插值函数和节点的图形

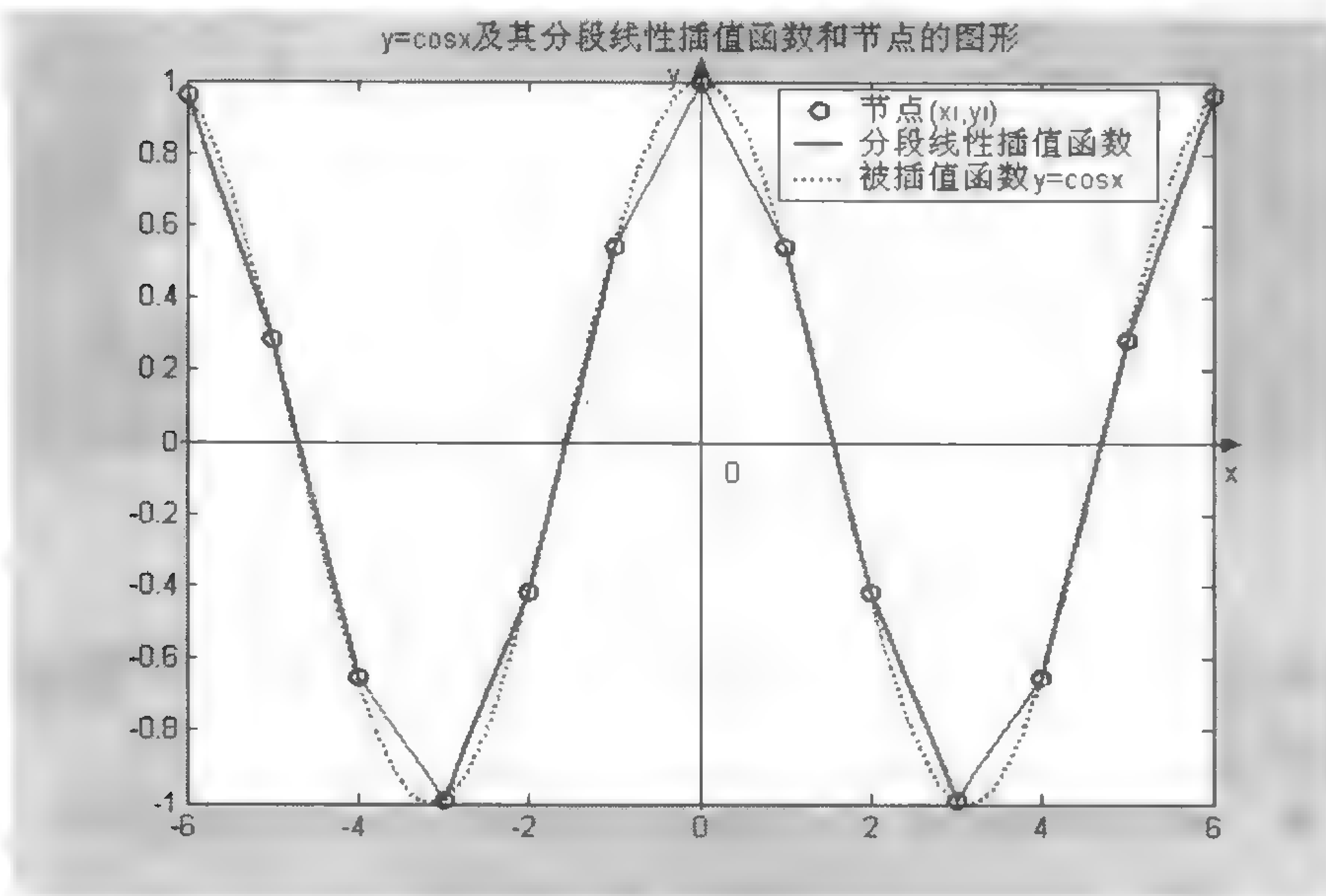


图 6－12 $y=\cos x$ 及其分段线性插值函数和节点的图形

6－13分别是在区间[－6,6]上被插值函数 $y=\cos x$(虚线)和其在节点(x_i,y_i)(其中 $x_i=-6+1.5i\ (i=0,1,2,\cdots 12)$)(小圆圈)处的分段线性插值函数(实线)和节点(小圆圈)的图形与分段三次埃尔米特插值函数(实线)和节点(小圆圈)的图形. 在节点附近分段线性插值函数与被插值函数的误差比分段三次埃尔米特插值函数的大. 并且 $y=\cos x$ 的分段三次埃尔米特插值函数的曲线几乎与 $y=\cos x$ 的曲线重合.

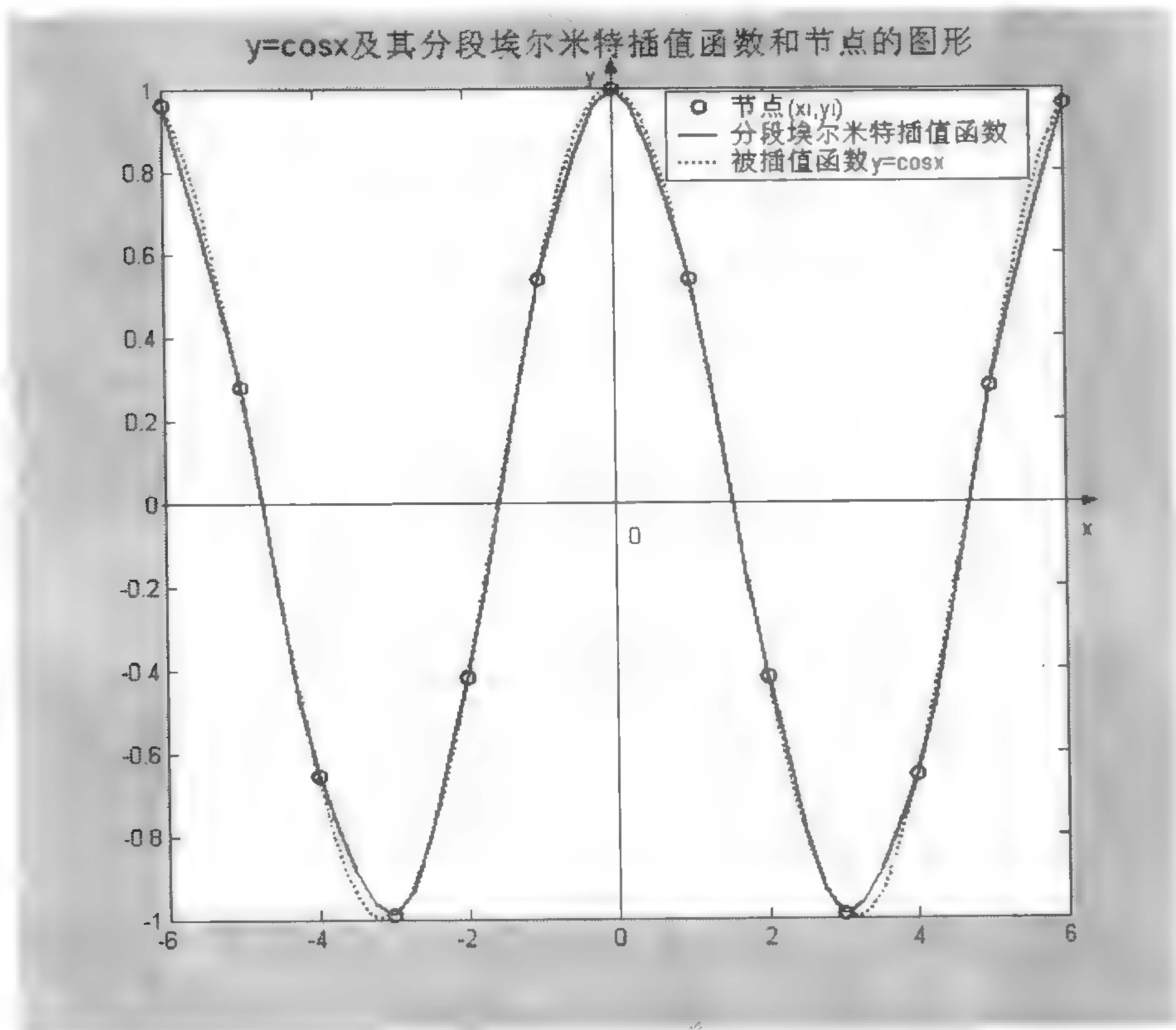


图 6－13 $y=\cos x$ 及其分段埃尔米特插值函数和节点的图形

例 6.6.1 设函数 $y=\dfrac{1}{1+25x^2}$定义在区间[－1,1]上,取 $n=10$,按等距节点构造分段三次埃尔米特插值函数 $H_{n,3}(x)$.

解 记节点的横坐标 $x_i=-1+ih, h=0.2, i=0,1,2,\cdots,10$,输入程序

```
>>xi = -1: 0.2: 1,yi =1./(1 +25 * x.^2),
yxi =  50./(1 +25.* x.^2).^2.* x
```

运行后输出节点的横坐标 x_i,纵坐标 y_i 和对应的导数值 y_i' 如下:

```
    xi =
       -1   -4/5   -3/5   -2/5   -1/5   0   1/5   2/5   3/5  4/5  1
    yi =
      1/26   1/17  1/10  1/5   1/2    1   1/2    1/5    1/10   1/
17   1/26
    yxi =
      25/338  40/289  3/10  4/5  5/2  0   -5/2   -4/5   -3/10
-40/289   -25/338
```

则分段三次埃尔米特插值函数为

$$H_{10,3}(x)=\{H_i(x)\mid H_i(x)=g_{i-1}(x)y_{i-1}+h_{i-1}(x)y'_{i-1}+g_i(x)y_i+h_i(x)y'_i,$$
$$x\in[x_{i-1},x_i],\ i=1,2,\cdots,10\},$$

其中 $H_{10,3}(x)$ 在每个子区间 $[x_{i-1},x_i]$ $(i=1,2,\cdots,10)$ 上的基函数为

$$g_{i-1}(x)=\left(1+2\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)\cdot\left(\frac{x-x_i}{x_{i-1}-x_i}\right)^2=(10x+13-2i)\cdot(5x+5-i)^2,$$

$$g_i(x)=\left(1+2\frac{x-x_i}{x_{i-1}-x_i}\right)\cdot\left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2=(-10x-9+2i)\cdot(5x+6-i)^2,$$

$$h_{i-1}(x)=(x-x_{i-1})\cdot\left(\frac{x-x_i}{x_{i-1}-x_i}\right)^2=(x+1.2-0.2i)\cdot(5x+5-i)^2,$$

$$h_i(x)=(x-x_i)\cdot\left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2=(x+1-0.2i)\cdot(5x+6-i)^2,$$

$$x\in[-1.2+0.2i,\ -1+0.2i].$$

例如，$H_{10,3}(x)$ 在子区间 $[-1,-0.8]$ 上的基函数为

$$g_0(x)=\left(1+2\frac{x-x_0}{x_1-x_0}\right)\cdot\left(\frac{x-x_1}{x_0-x_1}\right)^2=(10x+11)\cdot(5x+4)^2,$$

$$g_1(x)=\left(1+2\frac{x-x_1}{x_0-x_1}\right)\cdot\left(\frac{x-x_0}{x_1-x_0}\right)^2=-25(10x+7)\cdot(x+1)^2,$$

$$h_0(x)=(x-x_0)\cdot\left(\frac{x-x_1}{x_0-x_1}\right)^2=(x+1)\cdot(5x+4)^2,$$

$$h_1(x)=(x-x_1)\cdot\left(\frac{x-x_0}{x_1-x_0}\right)^2=5(5x+4)\cdot(x+1)^2.$$

故 $y=\dfrac{1}{1+25x^2}$ 在子区间 $[-1,-0.8]$ 上的分段三次埃尔米特插值函数为

$$H_1(x)=g_0(x)y_0+h_0(x)y'_0+g_1(x)y_1+h_1(x)y'_1,$$

即

$$H_1(x)=\frac{1}{26}(10x+11)\cdot(5x+4)^2+\frac{25}{338}(x+1)\cdot(5x+4)^2$$
$$-\frac{25}{17}(10x+7)\cdot(x+1)^2+\frac{200}{289}(5x+4)\cdot(x+1)^2.$$

6.6.2 分段埃尔米特插值的 MATLAB 程序

分段三次埃尔米特插值在 MATLAB 函数库中有现成的程序,在计算时直接调用即可,详细的调用方法如下:

调用格式一:`YI = interp1(X,Y,XI,'pchip')`

`interp1(X,Y,XI,'pchip')`命令的主要功能是计算函数 $\boldsymbol{Y}$ 在插值点向量 $\boldsymbol{XI}$ 的元素处的分段三次埃尔米特内插值所对应的向量 $\boldsymbol{YI}$.

(1) 如果输入的节点$(\boldsymbol{X}(i),\boldsymbol{Y}(i))$的横坐标向量 $\boldsymbol{X}$ 是 n 维,则纵坐标向量 $\boldsymbol{Y}$ 也应该是 n 维. 插值点向量 $\boldsymbol{XI}$ 是 m 维,则运行后输出的插值向量 $\boldsymbol{YI}$ 也是 m 维. 参考例 6.6.2(1)和例 6.6.4 等.

(2) 如果输入的节点$(\boldsymbol{X}(i),\boldsymbol{Y}(i))$的横坐标向量 $\boldsymbol{X}$ 是 n 维,纵坐标矩阵 $\boldsymbol{Y}_{n\times t}$的行数必须是 $\boldsymbol{X}$ 的维数 n,插值点向量 $\boldsymbol{XI}$ 是 m 维,则按矩阵 $\boldsymbol{Y}$ 的每列进行插值运算,运行后输出的插值矩阵 $\boldsymbol{YI}$ 是 m 行 t 列.

调用格式二:`YI = interp1(Y,XI,'pchip')`

`interp1(Y,XI'pchip')`命令与查表的作用相同.这表指的是$[\boldsymbol{X},\boldsymbol{Y}]$,要查在 $\boldsymbol{X}$ 的元素之间位置的 $\boldsymbol{XI}$ 的元素的分段三次埃尔米特内插值 $\boldsymbol{YI}$ 的元素,即返回值.

`interp1(Y,XI)`命令主要用于节点$(\boldsymbol{X}(i),\boldsymbol{Y}(i))$的横坐标向量 $\boldsymbol{X}$ 的元素是 1 到 n 的自然数. 由于 `interp1(Y,XI,'pchip')`是 `interp1(X,Y,XI,'pchip')`的特例,所以两者输入和输出的向量或矩阵的大小类似.即

(1) 如果输入的节点$(\boldsymbol{X}(i),\boldsymbol{Y}(i))$的横坐标向量 $\boldsymbol{X}$ 的元素是 1 到 n 的自然数,则纵坐标向量 $\boldsymbol{Y}$ 也应该是 n 维. 插值点向量 $\boldsymbol{XI}$ 是 m 维,则运行后输出的插值向量 $\boldsymbol{YI}$ 也是 m 维. 参考例 6.6.3(1).

(2) 如果输入的节点$(\boldsymbol{X}(i),\boldsymbol{Y}(i))$的横坐标向量 $\boldsymbol{X}$ 的元素是 1 到 n 的自然数,纵坐标矩阵 $\boldsymbol{Y}_{n\times t}$的行数必须是 $\boldsymbol{X}$ 的维数 n,插值点向量 $\boldsymbol{XI}$ 是 m 维,则按矩阵 $\boldsymbol{Y}$ 的每列进行插值运算,运行后输出的插值矩阵 $\boldsymbol{YI}$ 是 m 行 t 列.

例 6.6.2 给定节点$(\boldsymbol{X}(i),\boldsymbol{Y}(i))$的横坐标向量 $\boldsymbol{X}$,纵坐标向量或矩阵 $\boldsymbol{Y}$,插值点向量 $\boldsymbol{XI}$ 如下,计算分段三次埃尔米特插值向量 $\boldsymbol{YI}$.

(1) $\boldsymbol{X}=(-5,-3,-2,5)$,$\boldsymbol{Y}=(2,3,4,5)$,$\boldsymbol{XI}=1.375$;

(2) $\boldsymbol{X}=(-25,-31,-27,65)$，$\boldsymbol{Y}=\begin{pmatrix} 62 & 73 & 84 \\ -5 & -3 & -2 \\ -5/2 & -21/10 & -19/10 \\ -1 & 2 & 5 \end{pmatrix}$，$\boldsymbol{XI}$

是 -4 到 4 的整数.

解 (1) 在 MATLAB 工作窗口输入程序

```
>> X=[-5,-3,-2,5]; Y=[2,3,4,5];
XI=1.375; YI=interp1(X,Y,XI,'pchip')
```

运行后屏幕显示

```
YI =
    4.75171615692981
```

(2) 在 MATLAB 工作窗口输入程序

```
>>  X=[-25,-31,-27,65];
Y=[62,73,84;-5,-3,-2;-2.5,-2.1,-1.9;-1,2,5];
    XI=-4:4; YI=interp1(X,Y,XI,'pchip')
```

运行后屏幕显示

```
YI =
    61.19966666666667   72.09803703703703   82.99640740740740
    61.07980246913580   71.96295198902607   82.84610150891632
    60.94853086419753   71.81501097393689   82.68149108367626
    60.80533333333334   71.65362962962963   82.50192592592593
    60.64969135802469   71.47822359396433   82.30675582990398
    60.48108641975309   71.28820850480109   82.09533058984911
    60.29900000000000   71.08300000000000   81.86700000000000
    60.10291358024691   70.86201371742112   81.62111385459534
    59.89230864197531   70.62466529492455   81.35702194787380
```

例 6.6.3 给定节点$(\boldsymbol{X}(i),\boldsymbol{Y}(i))$的横坐标向量 $\boldsymbol{X}$ 的元素是 1 到 4 的正整数，纵坐标向量或矩阵 $\boldsymbol{Y}$，插值点向量 $\boldsymbol{XI}$ 与例 6.6.2 相同，计算分段三次埃尔米特插值向量 $\boldsymbol{YI}$.

解 (1) 在 MATLAB 工作窗口输入程序

```
>> Y=[2,3,4,5]; XI=1.375; YI=interp1(Y,XI,'pchip')
```

运行后屏幕显示

```
YI =
    2.37500000000000
```

(2) 在 MATLAB 工作窗口输入程序

```
>> Y=[62,73,84;-5,-3,-2;-2.5,-2.1,-1.9;-1,2,5];
```

```
XI = -4:4; YI = interp1(Y,XI,'pchip')
```

运行后屏幕显示

```
YI =
    1.0e+003 *
  -3.39800000000000   -4.02600000000000   -4.63700000000000
  -1.55500000000000   -1.85800000000000   -2.14700000000000
  -0.48100000000000   -0.58940000000000   -0.68760000000000
   0.01750000000000    0.00510000000000   -0.00110000000000
   0.13400000000000    0.15080000000000    0.17020000000000
   0.06200000000000    0.07300000000000    0.08400000000000
  -0.00500000000000   -0.00300000000000    0.00200000000000
  -0.00250000000000   -0.00210000000000   -0.00190000000000
  -0.00100000000000    0.00200000000000    0.00500000000000
```

例 6.6.4 给定节点$(\boldsymbol{X}(i),\boldsymbol{Y}(i))$的横坐标向量$\boldsymbol{X}$的元素是0到10的整数,由$y=\sin x$确定纵坐标向量$\boldsymbol{Y}$,插值点向量$\boldsymbol{XI}$的元素是首项$a=0$,末项$b=10$,公差$h=0.5$的等差数列,计算分段三次埃尔米特插值向量$\boldsymbol{YI}$.

解 在 MATLAB 工作窗口输入程序

```
>> X=0:10; Y=sin(X); XI=0:.5:10; YI=interp1(X,Y,XI,'pchip')
```

运行后屏幕显示

```
YI =
    Columns 1 through 7
         0    0.5586    0.8415    0.8911    0.9093    0.6287    0.1411
    Columns 8 through 14
      -0.3701   -0.7568   -0.8991   -0.9589   -0.7176   -0.2794    0.2259
    Columns 15 through 21
       0.6570    0.8845    0.9894    0.7907    0.4121   -0.0127   -0.5440
```

例 6.6.5 试用 MATLAB 程序计算例 6.6.1 中在各小区间中点处分段三次埃尔米特插值$H_n(x_{i+1/2})$及其相对误差.

解 (1) 记节点的横坐标$x_i=-1+ih,h=0.2,i=0,1,2,\cdots,10$,插值点$x_{i+\frac{1}{2}}=\dfrac{1}{2}(x_i+x_{i+1}),i=0,1,2,\cdots,9$. 则

① 分段三次埃尔米特插值函数为

$$H_{10,3}(x)=\{H_i(x)\mid H_i(x)=g_{i-1}(x)y_{i-1}+h_{i-1}(x)y_{i-1}'+g_i(x)y_i+h_i(x)y_i',x\in[x_{i-1},x_i],i=1,2,\cdots,10\},$$

其中$H_{10,3}(x)$在每个子区间$[x_{i-1},x_i](i=1,2,\cdots,n)$上的基函数见例 6.6.1.

② $H_{10,3}(x)$在小区间$[x_i,x_{i+1}],i=0,1,2,\cdots,9$ 中点$x_{i+\frac{1}{2}}=\dfrac{1}{2}(x_i+x_{i+1})$,

$i=0,1,2,\cdots,9$处的插值

$$H_{10,3}(x_{i+\frac{1}{2}})=\frac{1}{2}[f(x_i)+f(x_{i+1})],\ i=0,1,2,\cdots,9.$$

③ 插值 $H_{10,3}(x_{i+\frac{1}{2}})$ 的相对误差公式可以表示为

$$|R_{10,3}(x_{i+\frac{1}{2}})|=\left|\frac{f(x_{i+\frac{1}{2}})-H_{10}(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right|.$$

（2）下面用 MATLAB 程序计算各小区间中点处插值 $H_{10,3}(x_{i+\frac{1}{2}})$ 及其相对误差 $|R_{10,3}(x_{i+\frac{1}{2}})|$ 的值. 在 MATLAB 工作窗口输入程序

```
>> h = 0.2;x0 = -1:h:1;y0 =1./(1 +25.*x0.^2); xi = -0.9:h:0.9;
   fi =1./(1 +25.*xi.^2); yi = interp1(x0,y0,xi,'pchip');
   Ri = abs((fi -yi)./fi); xi,fi,yi,Ri,i =[xi',fi',yi',Ri']
```

运行后屏幕显示各小区间中点 $x_i=x_{i+\frac{1}{2}}$ 处的函数值 $f_i=f(x_{i+\frac{1}{2}})$，插值 $s_i=H_{10,3}(x_{i+\frac{1}{2}})$，相对误差值 $R_i=\left|\frac{f(x_{i+\frac{1}{2}})-H_{10,3}(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right|$，经过整理后列入表 6－5 中：

表 6－5 例 6.6.5 的各小区间中点 x_i 处的函数值 f_i，插值 s_i 和相对误差值 R_i

i	$x_{i+\frac{1}{2}}=\frac{1}{2}(x_i+x_{i+1})$	$f(x_{i+\frac{1}{2}})$	$H_{10,3}(x_{i+\frac{1}{2}})$	$\left\|\frac{f(x_{i+\frac{1}{2}})-H_{10,3}(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right\|$
0	-0.900 000 000 000 00	0.047 058 823 529 41	0.046 480 735 959 54	0.012 284 360 859 73
1	-0.700 000 000 000 00	0.075 471 698 113 21	0.075 526 239 907 73	0.000 722 678 777 39
2	-0.500 000 000 000 00	0.137 931 034 482 76	0.138 541 666 666 67	0.004 427 083 333 33
3	-0.300 000 000 000 00	0.307 692 307 692 31	0.321 875 000 000 00	0.046 093 750 000 00
4	-0.100 000 000 000 00	0.800 000 000 000 00	0.796 875 000 000 00	0.003 906 250 000 00
5	0.100 000 000 000 00	0.800 000 000 000 00	0.796 875 000 000 00	0.003 906 250 000 00
6	0.300 000 000 000 00	0.307 692 307 692 31	0.321 875 000 000 00	0.046 093 750 000 00
7	0.500 000 000 000 00	0.137 931 034 482 76	0.138 541 666 666 67	0.004 427 083 333 33
8	0.700 000 000 000 00	0.075 471 698 113 21	0.075 526 239 907 73	0.000 722 678 777 39
9	0.900 000 000 000 00	0.047 058 823 529 41	0.046 480 735 959 54	0.012 284 360 859 73

6.6.3 作有关分段埃尔米特插值图形的 MATLAB 程序

如果能在同一个坐标系中作出在插值区间 $[a,b]$ 上的节点 $(x_i,f(x_i))$，被插值函数 $f(x)$，$f(x)$ 的分段埃尔米特插值函数 $H_{n,3}(x)$ 和插值点 $(x_j,H_{n,3}(x_j))$ 等，则更有利于我们进一步直观地研究我们关心的问题. 为此编写了下面的 MATLAB 程序.

作有关分段埃尔米特插值图形的 MATLAB 主程序

输入的量：$n+1$ 个节点$(x_i, f(x_i))$ $(i=1,2,\cdots,n+1)$横坐标向量 $\boldsymbol{x}_0$，纵坐标向量 $\boldsymbol{y}_0$，插值点$(x_j, H_{n,3}(x_j))$横坐标向量 $\boldsymbol{x}_i$ 和函数 $y=f(x)$值.

输出的量：插值 $H=H_{n,3}(x_j)$.

输出的图形：在插值区间$[a,b]$上的节点$(x_i, f(x_i))$，被插值函数 $f(x)$，$f(x)$的分段埃尔米特插值函数 $H_{n,3}(x)$和插值点$(x_j, H_{n,3}(x_j))$.

```
function H = hermitetx(x0,y0,xi,x,y)
H = interp1(x0,y0,xi,'pchip'); Hn = interp1(x0,y0,x,'pchip');
plot(x0,y0,'o',x,Hn,'-',xi,H,'*',x,y,'-.')
legend('节点(xi,yi)', '分段埃尔米特插值函数','插值点(x,H)','被插值函
数 y')
```

我们也可以直接在 MATLAB 工作窗口编程序，例如，

```
>> x0 = -6:6; y0 = sin(x0); xi = -6:.25:6;
yi = interp1(x0,y0,xi,'pchip');
x = -6:0.001:6; y = sin(x); plot(x0,y0,'o',xi,yi,x,y,':'),
legend('节点(xi,yi)','分段埃尔米特插值函数','y = sin x 的函数')
>> x0 = -6:6; y0 = cos(x0);
xi = -6:.25:6;yi = interp1(x0,y0,xi,'pchip');
x = -6:0.001:6;y = cos(x); plot(x0,y0,'o',xi,yi,x,y,':'),
legend('节点(xi,yi)','分段埃尔米特插值函数','y = cosx 的函数')
```

例 6.6.6 设函数 $f(x)=\dfrac{1}{1+x^2}$定义在区间$[-5,5]$上，节点$(X(i), f(X(i)))$的横坐标向量 $\boldsymbol{X}$ 的元素是首项 $a=-5$，末项 $b=5$，公差 $h=1.5$ 的等差数列，构造三次分段埃尔米特插值函数 $H_{n,3}(x)$. 把区间$[-5.5]$分成 20 等份，构成 20 个小区间，用 MATLAB 程序计算各小区间中点 x_i 处 $H_{n,3}(x)$的值，并作出节点、插值点、$f(x)$和 $H_{n,3}(x)$的图形.

解 记等分点 $x_i=-5+ih, h=0.5, i=0,1,2,\cdots,20$，插值点 $x_{i+\frac{1}{2}}=\dfrac{1}{2}(x_i+x_{i+1})$，$i=0,1,2,\cdots,19$. 在 MATLAB 工作窗口输入程序

```
>> x0 = -5:1.5:5;
y0 =1./(1 +x0.^2); x1 = -4.75:0.5:4.75;
x = -5:0.001:5; y =1./(1 +x.^2); H =hermitetx(x0,y0,x1,x,y)
title('函数 y =1/(1 +x^2)及其分段埃尔米特插值函数,插值,节点(xi,yi)
的图形')
```

运行后屏幕显示各小区间中点 x_i 处 $H_{n,3}(x)$的值，节点、插值点、$f(x)$和 $H_{n,3}(x)$的图形（见图 6－14）如下

```
H =   Columns 1 through 7
    0.0399  0.0498  0.0661  0.0865  0.1191  0.1682  0.2683
  Columns 8 through 14
    0.5258  0.7603  0.7854  0.6910  0.5602  0.4382  0.2942
  Columns 15 through 20
    0.1722  0.1170  0.0821  0.0617  0.0619  0.0892
```

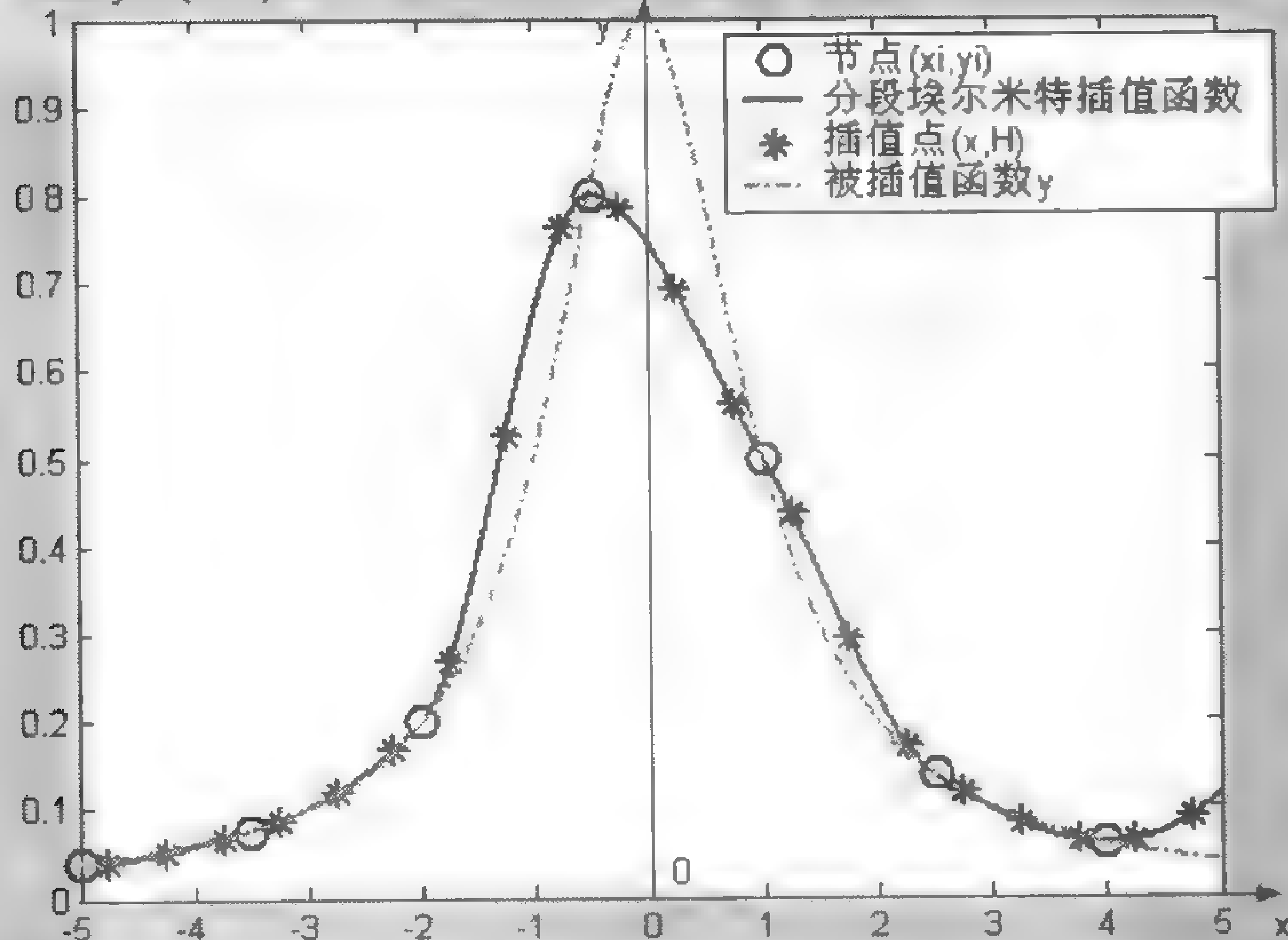


图 6-14 函数 $f(x)=\frac{1}{1+x^2}$ 及其分段埃尔米特插值函数，插值，节点的图形

请读者分别取公差 $h=1,0.5,0.05,0.01,0.001$，观察函数 $f(x)=\frac{1}{1+x^2}$ 及分段埃尔米特插值函数的图形，说明误差有何变化.

例 6.6.7 设函数 $f(x)=0.5x-\cos x$ 定义在区间 $[-\pi,\pi]$ 上，取 $n=7$，按等距节点构造分段埃尔米特插值函数 $H_{7,3}(x)$，用 MATLAB 程序计算各小区间中点 x_i 处 $H_{7,3}(x)$ 的值，作出节点、插值点、$f(x)$ 和 $H_{7,3}(x)$ 的图形.

解 记节点的横坐标 $x_i=-\pi+ih,h=2\pi/7,i=0,1,2,\cdots,7$，插值点 $x_{i+\frac{1}{2}}=\frac{1}{2}(x_i+x_{i+1}),i=0,1,2,\cdots,6$. 在 MATLAB 工作窗口输入程序

```
>>  h=2*pi/7; x0=-pi:h:pi;
y0=0.5.*x0-cos(x0); xi=-pi+h/2:h:pi-h/2;
```

```
        b = max(x0); a = min(x0); x = a:0.001:b;
        y = 0.5.*x - cos(x); H = hermitetx(x0,y0,xi,x,y)
        title('函数 y = 0.5x - cos(x)及其分段埃尔米特插值函数,插值,节点(xi,
yi) 的图形')
```

运行后屏幕显示各小区间中点 x_i 处 $H_{7,3}(x)$ 的值，节点、插值点、$f(x)$ 和 $H_{7,3}(x)$ 的图形（见图 6-15）如下：

```
        H =   -0.5075   -0.6607   -1.0469   -0.9812   -0.1834   1.1227
2.2103
```

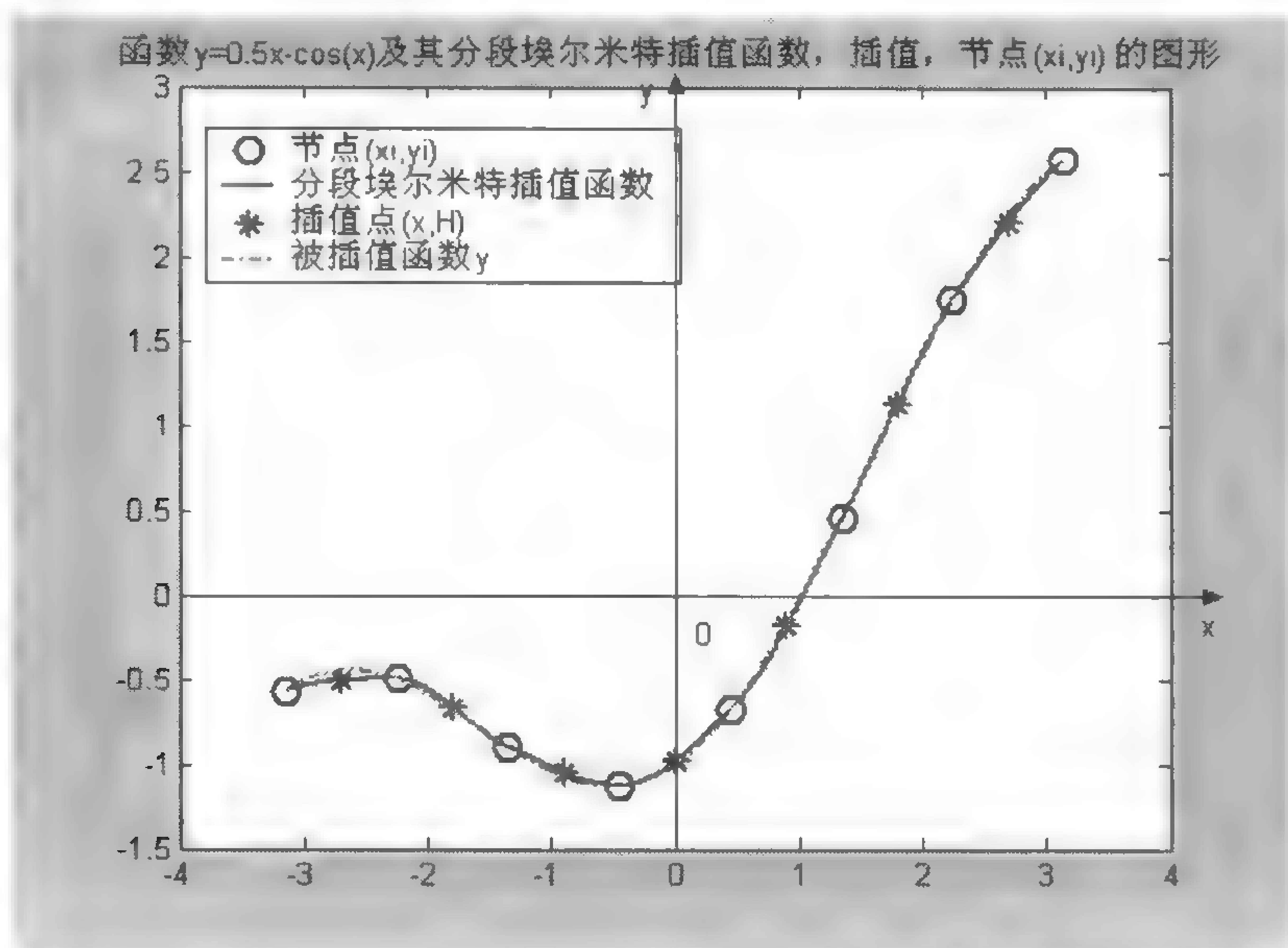


图 6-15 $y=0.5x-\cos x$ 及其分段埃尔米特插值函数，插值，节点的图形

6.6.4 用 MATLAB 计算有关分段埃尔米特插值的误差

定理 6.10 如果被插值函数 $f(x)$ 在插值区间 $[a,b]$ 上具有 4 阶连续导数，$[a,b]$ 上的 $n+1$ 个节点 (x_i,y_i) $(i=0,1,2,\cdots,n)$ 满足 $a=x_0<x_1<x_2<\cdots<x_n=b$，$H_{n,3}(x)$ 是定义 6.4 所定义的 $f(x)$ 在 $[a,b]$ 上的分段埃尔米特插值函数，则

（1）被插值函数 $f(x)$ 在由相邻节点 (x_i,y_i) $(i=0,1,2,\cdots,n)$ 构成的子区间 $[x_{i-1},x_i]$ 上的 $H_{n,3}(x)$ 的余项为

$$R_i(x)=f(x)-H_{n,3}(x)=\frac{f^{(4)}(\xi_i)}{4!}(x-x_{i-1})^2(x-x_i)^2,x,\xi_i\in(x_{i-1},x_i),\tag{6.53}$$

误差公式可以表示为

$$|R_i(x)| = |f(x) - H_{n,3}(x)| \leqslant \frac{(x_i - x_{i-1})^4}{384} \max_{x_{i-1} \leqslant x \leqslant x_i} |f^{(4)}(x)|. \quad (6.54)$$

(2) $H_{n,3}(x)$在插值区间$[a,b]$上的误差公式可以表示为

$$|R_{n,3}(x)| = |f(x) - H_{n,3}(x)| \leqslant \frac{\max\limits_{1 \leqslant i \leqslant n} |x_i - x_{i-1}|^4}{384} \max_{a \leqslant x \leqslant b} |f^{(4)}(x)|. \quad (6.55)$$

且 $H_{n,3}(x)$在$[a,b]$上一致收敛到$f(x)$.

证明 (1) 根据定理 6.6 知,在子区间$[x_{i-1},x_i]$上的 $H_{n,3}(x)$是三次埃尔米特插值多项式,且余项为

$$R_i(x) = f(x) - H_{n,3}(x) = \frac{f^{(4)}(\xi)}{4!}(x - x_{i-1})^2(x - x_i)^2, \xi \in (x_{i-1}, x_i),$$

所以

$$|R_i(x)| = \left|\frac{f^{(4)}(\xi)}{4!}(x - x_{i-1})^2(x - x_i)^2\right| \leqslant \left|\frac{f^{(4)}(\xi)}{4!}\left(\frac{x_i - x_{i-1}}{2}\right)^2 \cdot \left(\frac{x_{i-1} - x_i}{2}\right)^2\right|.$$

$$\leqslant \frac{(x_i - x_{i-1})^4}{384} \max_{x_{i-1} \leqslant x \leqslant x_i} |f^{(4)}(x)|.$$

(2) $H_{n,3}(x)$在插值区间$[a,b]$上的误差为

$$|R_{n,3}(x)| = \max_{1 \leqslant i \leqslant n}\{|R_i(x)|\} \leqslant \frac{\max\limits_{1 \leqslant i \leqslant n} |x_i - x_{i-1}|^4}{384} \max_{a \leqslant x \leqslant b} |f^{(4)}(x)|.$$

因为等距节点所构造的子区间$[x_{i-1},x_i]$的长度相等,所以$\max\limits_{1 \leqslant i \leqslant n} |x_i - x_{i-1}|^4 = |x_i - x_{i-1}|^4 = h^4$. 由定理 6.10 可以得到下面的推论.

推论 6.2 如果被插值函数$f(x)$在插值区间$[a,b]$上具有 4 阶连续导数,$[a,b]$上的 $n+1$ 个节点(x_i,y_i) $(i=0,1,2,\cdots,n)$的横坐标是等距离点,即 $x_i = a + h \cdot i$,其中 $h = \frac{b-a}{n}$,$i=0,1,2,\cdots,n$,$H_{n,3}(x)$是定义 6.4 所定义的$f(x)$在$[a,b]$上的分段埃尔米特插值函数,则

(1) 被插值函数$f(x)$在由相邻节点(x_i,y_i) $(i=0,1,2,\cdots,n)$构成的子区间$[x_{i-1},x_i]$上的 $H_{n,3}(x)$的余项为

$$R_i(x) = f(x) - H_{n,3}(x) = \frac{f^{(4)}(\xi_i)}{4!}(x - x_{i-1})^2(x - x_i)^2, x, \xi_i \in (x_{i-1}, x_i), \quad (6.56)$$

误差公式可以表示为

$$|R_i(x)| = |f(x) - H_{n,3}(x)| \leqslant \frac{h^4}{384} \max_{x_{i-1} \leqslant x \leqslant x_i} |f^{(4)}(x)|. \quad (6.57)$$

(2) $H_{n,3}(x)$在插值区间$[a,b]$上的误差公式可以表示为

$$|R_{n,3}(x)| = |f(x) - H_{n,3}(x)| \leqslant \frac{h^4}{384} \max_{a \leqslant x \leqslant b} |f^{(4)}(x)|. \quad (6.58)$$

例 6.6.8 设函数$f(x)=\frac{1}{1+25x^2}$定义在区间[-1,1]上,取 $n=10$,按等距节点构造分段埃尔米特插值函数 $H_{n,3}(x)$,用 MATLAB 程序在[-1,1]上计算$\max\limits_{-1\leq x\leq 1}|f^{(4)}(x)|$和 $H_{n,3}(x)$的误差公式和误差限.

解 (1) 节点的横坐标和插值点等取值与例 6.6.1 相同. 即 $H_{n,3}(x)$在区间[-1,1]上的误差公式可以表示为

$$|R_{10,3}(x)|=|f(x)-H_{10,3}(x)|\leq\frac{1}{240\ 000}\max_{-1\leq x\leq 1}|f^{(4)}(x)|.$$

(2) 在 MATLAB 工作窗口输入程序

```
>>syms  x
y=1/(1+25*x^2); yxxxx=diff(y,x,4),
```

运行后输出$f(x)$的 4 阶导数(略).

(3) 在 MATLAB 工作窗口输入程序

```
>>syms h,x= -1:0.0001:1;
yxxxx=150000000./(1+25.*x.^2).^5.*x.^4 -4500000./(1+25.*
      x.^2).^4.*x.^2 +15000./(1+25.*x.^2).^3;
myxxxx=max(yxxxx), R=(h^4)* abs(myxxxx/384)
```

运行后输出$f(x)$的 4 阶导数在区间[-1,1]上绝对值的最大值 *myxxxx* 和 $H_{n,3}(x)$在区间[-1,1]上的误差公式 R 为

```
myxxxx =                    R =
        15000                   625/16*h^4
```

(4) 在 MATLAB 工作窗口输入程序

```
>>h=0.2; R=625/16*h^4
```

运行后输出误差限为

```
R =
   0.06250000000000
```

例 6.6.9 设函数$f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$定义在区间[-π,π]上,取 $n=9$,按等距节点构造分段埃尔米特插值函数 $H_{n,3}(x)$.

(1) 用 MATLAB 程序计算各小区间中点 x_i 处 $H_{n,3}(x)$的值,作出节点、插值点、$f(x)$和 $H_{n,3}(x)$的图形;

(2) 用 MATLAB 程序计算各小区间中点处 $H_{n,3}(x)$的值及其相对误差;

(3) 用 MATLAB 程序求$\max\limits_{-\pi\leq x\leq\pi}|f^{(4)}(x)|$和 $H_{n,3}(x)$在区间[-π,π]上的误差公式和各插值的误差限.

解 (1) 记节点的横坐标 $x_i=-\pi+ih,h=2\pi/9,i=0,1,2,\cdots,9$,插值点

$x_{i+\frac{1}{2}}=\frac{1}{2}(x_i+x_{i+1}), i=0,1,2,\cdots,8$. 在 MATLAB 工作窗口输入程序

```
>> h=2*pi/9; x0=-pi:h:pi;
y0=tan(cos((3^(1/2)+sin(2*x0))./(3+4*x0.^2)));
xi=-pi+h/2:h:pi-h/2;
fi=tan(cos((3^(1/2)+sin(2*xi))./(3+4*xi.^2)));
b=max(x0); a=min(x0); x=a:0.001:b;
y=tan(cos((3^(1/2)+sin(2.*x))./(3+4*x.^2)));
Hi=hermitetx(x0,y0,xi,x,y);
Ri=abs((fi-yi)./fi); xi,fi,Hi,Ri,i=[xi',fi',Hi',Ri']
title('函数y=tan(cos((sqrt(3)+sin(2x))/(3+4x^2)))及其分段埃
尔米特插值函数,插值,节点(xi,yi) 的图形')
```

运行后屏幕显示各小区间中点 $x_i=x_{i+\frac{1}{2}}$ 处的函数值 $f_i=f(x_{i+\frac{1}{2}})$，插值 $H_i=H_{n,3}(x_{i+\frac{1}{2}})$，相对误差值 $R_i=\left|\frac{f(x_{i+\frac{1}{2}})-H_{n,3}(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right|$，并且作出节点、插值点、$f(x)$ 和 $H_{n,3}(x)$ 的图形（见图 6－16），经整理后填入表 6－6.

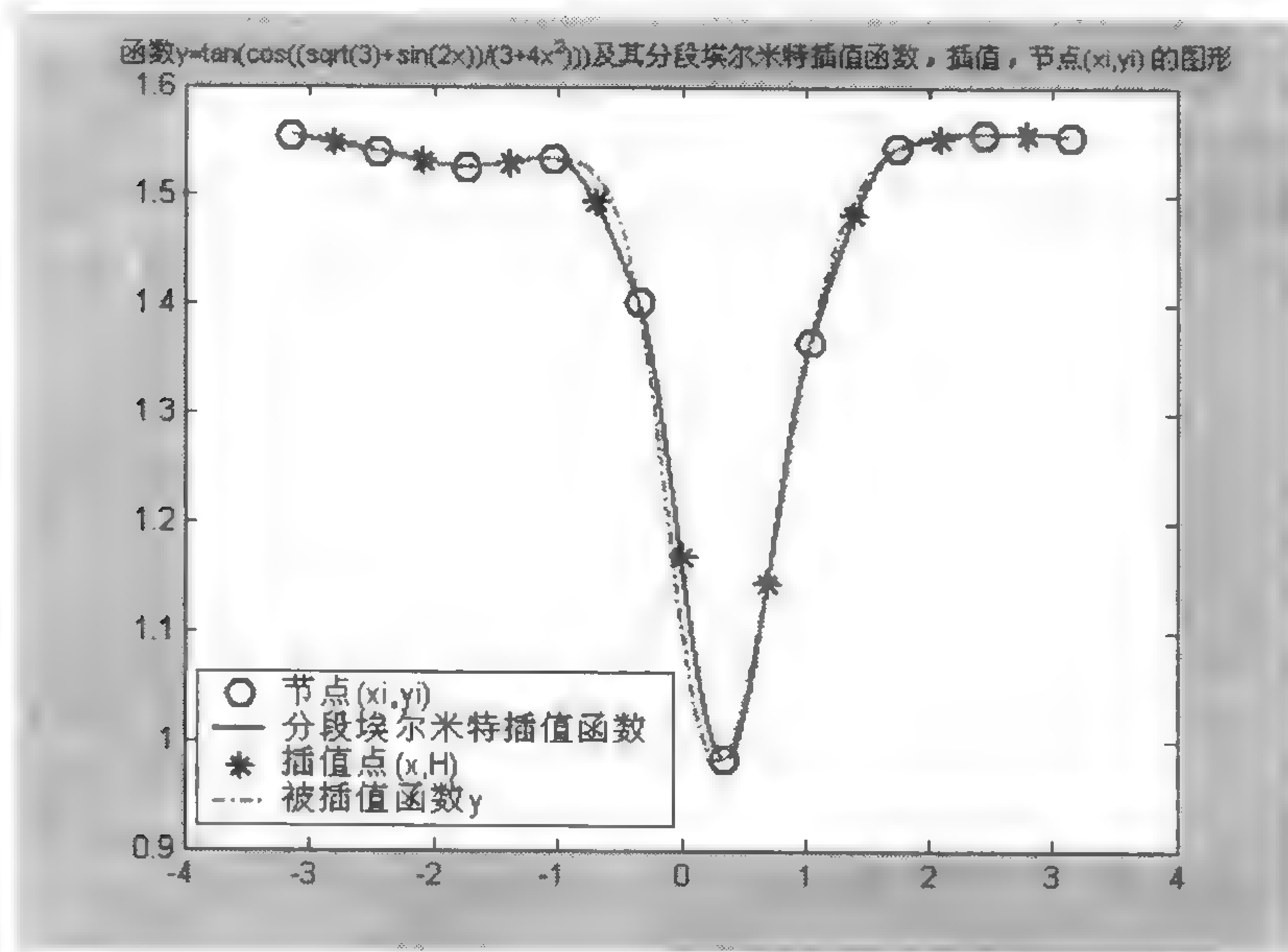


图 6－16　$f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$及其分段埃尔米特插值函数，插值，节点的图形

表 6－6 例 6.6.9 的小区间中点 x_i 处的函数值 f_i，插值 H_i，相对误差值 R_i

i	$x_{i+\frac{1}{2}}=\frac{1}{2}(x_i+x_{i+1})$	$f(x_{i+\frac{1}{2}})$	$H_{n,3}(x_{i+\frac{1}{2}})$	$\left\|\frac{f(x_{i+\frac{1}{2}})-H_{n,3}(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right\|$
0	−2.792 526 803 190 93	1.549 179 746 366 19	1.547 265 628 459 29	0.001 235 568 636 49
1	−2.094 395 102 393 20	1.530 392 391 872 18	1.531 224 342 982 60	0.000 543 619 476 18
2	−1.396 263 401 595 46	1.529 424 914 525 70	1.530 048 627 687 94	0.000 407 808 945 91
3	−0.698 131 700 797 73	1.519 120 343 743 33	1.493 612 317 915 46	0.016 791 313 428 81
4	0	1.110 956 050 058 38	1.168 255 711 383 02	0.051 576 893 002 77
5	0.698 131 700 797 73	1.145 460 398 956 37	1.144 189 914 880 54	0.001 109 147 096 64
6	1.396 263 401 595 46	1.496 160 173 406 94	1.481 729 818 014 74	0.009 644 926 825 82
7	2.094 395 102 393 20	1.554 369 358 742 26	1.552 633 897 250 18	0.001 116 505 213 08
8	2.792 526 803 190 93	1.555 671 074 430 44	1.555 895 063 017 64	0.000 143 981 970 79

（2）在 MATLAB 工作窗口输入程序

```
>> syms  x
y=tan(cos((3^(1/2)+sin(2*x))/(3+4*x^2))); yxxxx=diff(y,
x,4),% simple(yxxxx)
```

运行后屏幕显示函数 $f(x)$ 的 4 阶导数 $f^{(4)}(x)$，然后将输出的 $f^{(4)}(x)$ 编程求 $\max\limits_{-\pi\leq x\leq\pi}|f^{(4)}(x)|$ 和 $H_{n,3}(x)$ 及其在区间 $[-\pi,\pi]$ 上的误差限的 MATLAB 程序如下：

```
>>syms h,x=-pi:0.0001:pi;
yxxxx=-12.*(1.+tan(cos((3.^(1./2)+sin(2.*x))./(3.+4.*
x.^2))).^2).^2.*sin((3.^(1./2)+sin(2.*x))./(3.+4.*x.^2)).^3.*(2.
*cos(2.*x)./(3.+4.*x.^2)-8.*(3.^(1./2)+sin(2.*x))./(3.+4.*x.^
2).^2.*x).^2.*(-4.*sin(2.*x)./(3.+4.*x.^2)-32.*cos(2.*x)./(3.
+4.*x.^2).^2.*x+128.*(3.^(1./2)+sin(2.*x))./(3.+4.*x.^2).^3.*
x.^2.-8.*(3.^(1./2)+sin(2.*x))./(3.+4.*x.^2).^2)+16.*(1.+tan
(cos((3.^(1./2)+sin(2.*x))./(3.+4.*x.^2))).^2).^2.*sin((3.^(1./
2)+sin(2.*x))./(3.+4.*x.^2)).^4.*(2.*cos(2.*x)./(3.+4.*x.^2)-
8.*(3.^(1./2)+sin(2.*x))./(3.+4.*x.^2).^2.*x).^4.*tan(cos((3.^
(1./2)+sin(2.*x))./(3.+4.*x.^2)))+8.*tan(cos((3.^(1./2)+sin(2.
*x))./(3.+4.*x.^2))).^3.*(1.+tan(cos((3.^(1./2)+sin(2.*x))./(3.
+4.*x.^2))).^2).*sin((3.^(1./2)+sin(2.*x))./(3+4.*x.^2)).^4.*
(2.*cos(2.*x)./(3+4.*x.^2)-8.*(3.^(1./2)+sin(2.*x))./(3.+4.*
```

```
x.^2).^2.*x).^4 -8.*tan(cos((3.^(1./2) +sin(2.*x))./(3.+4.*x.^
2))).*(1.+tan(cos((3.^(1./2) +sin(2.*x))./(3.+4*x.^2))).^2).*sin
((3.^(1./2) +sin(2.*x))./(3.+4.*x.^2)).^2.*(2.*cos(2.*x)./(3.+
4.*x.^2) -8*(3.^(1./2) +sin(2.*x))./(3.+4*x.^2).^2.*x).^4 +6.*(1
+tan(cos((3.^(1./2) +sin(2.*x))./(3 +4.*x.^2))).^2).*sin((3.^(1./
2) +sin(2.*x))./(3 +4.*x.^2)).*(2.*cos(2.*x)./(3 +4.*x.^2) -8.*
(3.^(1./2) +sin(2.*x))./(3 +4.*x.^2).^2.*x).^2.*( -4.*sin(2.*x)./
(3 +4.*x.^2) -32.*cos(2.*x)./(3 +4.*x.^2).^2.*x +128.*(3.^(1./2)
+sin(2.*x))./(3 +4.*x.^2).^3.*x.^2 -8.*(3.^(1./2) +sin(2.*x))./
(3 +4.*x.^2).^2) +(1 +tan(cos((3.^(1./2) +sin(2.*x))./(3 +4.*x.^
2))).^2).*cos((3.^(1./2) +sin(2.*x))./(3 +4.*x.^2)).*(2.*cos(2.*
x)./(3 +4.*x.^2) -8.*(3.^(1./2) +sin(2.*x))./(3 +4.*x.^2).^2.*x).^
4 -3.*(1 +tan(cos((3.^(1./2) +sin(2.*x))./(3 +4.*x.^2))).^2).*cos
((3.^(1./2) +sin(2.*x))./(3 +4.*x.^2)).*( -4.*sin(2.*x)./(3 +4.*
x.^2) -32.*cos(2.*x)./(3 +4.*x.^2).^2.*x +128.*(3.^(1./2) +sin(2.
*x))./(3 +4.*x.^2).^3.*x.^2 -8.*(3.^(1./2) +sin(2.*x))./(3 +4.*x.
^2).^2).^2 -4.*(1 +tan(cos((3.^(1./2) +sin(2.*x))./(3 +4.*x.^2))).^
2).*cos((3.^(1./2) +sin(2.*x))./(3 +4.*x.^2)).*(2.*cos(2.*x)./(3
+4.*x.^2) -8.*(3.^(1./2) +sin(2.*x))./(3 +4.*x.^2).^2.*x).*( -8.
*cos(2.*x)./(3 +4.*x.^2) +96.*sin(2.*x)./(3 +4.*x.^2).^2.*x +
768.*cos(2.*x)./(3 +4.*x.^2).^3.*x.^2 -48.*cos(2.*x)./(3 +4.*x.^
2).^2 -3072.*(3.^(1./2) +sin(2.*x))./(3 +4.*x.^2).^4.*x.^3 +384.*
(3.^(1./2) +sin(2.*x))./(3 +4.*x.^2).^3.*x) -(1 +tan(cos((3.^(1./
2) +sin(2.*x))./(3 +4.*x.^2))).^2).*sin((3.^(1./2) +sin(2.*x))./
(3 +4.*x.^2)).*(16.*sin(2.*x)./(3 +4.*x.^2) +256.*cos(2.*x)./(3
+4.*x.^2).^2.*x -3072.*sin(2.*x)./(3 +4.*x.^2).^3.*x.^2 +192.*
sin(2.*x)./(3 +4.*x.^2).^2 -24576.*cos(2.*x)./(3 +4.*x.^2).^4.*x.
^3 +3072.*cos(2.*x)./(3 +4.*x.^2).^3.*x +98304.*(3.^(1./2) +sin(2.
*x))./(3 +4.*x.^2).^5.*x.^4 -18432.*(3.^(1./2) +sin(2.*x))./(3 +
4.*x.^2).^4.*x.^2 +384.*(3.^(1./2) +sin(2.*x))./(3 +4.*x.^2).^3) -
12.*(1 +tan(cos((3.^(1./2) +sin(2.*x))./(3 +4.*x.^2))).^2).^2*sin
((3.^(1./2) +sin(2.*x))./(3 +4.*x.^2)).^2.*(2.*cos(2.*x)./(3 +4.
*x.^2) -8.*(3.^(1./2) +sin(2.*x))./(3 +4.*x.^2).^2.*x).^4.*cos
((3.^(1./2) +sin(2.*x))./(3 +4.*x.^2)) -24.*tan(cos((3.^(1./2) +
sin(2.*x))./(3 +4.*x.^2))).^2.*(1 +tan(cos((3.^(1./2) +sin(2.*
x))./(3 +4.*x.^2))).^2).*sin((3.^(1./2) +sin(2.*x))./(3 +4.*x.^
2)).^3.*(2.*cos(2.*x)./(3 +4.*x.^2) -8.*(3.^(1./2) +sin(2.*x))./
(3 +4.*x.^2).^2.*x).^2.*( -4.*sin(2.*x)./(3 +4.*x.^2) -32.*cos(2.
```

```
*x)./(3+4.*x.^2).^2.*x+128.*(3.^(1./2)+sin(2.*x))./(3+4.*x.^
2).^3.*x.^2-8.*(3.^(1./2)+sin(2.*x))./(3+4.*x.^2).^2)-24.*tan
(cos((3.^(1./2)+sin(2.*x))./(3+4.*x.^2))).^2.*(1+tan(cos((3.^
(1./2)+sin(2.*x))./(3+4.*x.^2))).^2).*sin((3.^(1./2)+sin(2.*
x))./(3+4.*x.^2)).^2.*(2.*cos(2.*x)./(3+4.*x.^2)-8.*(3.^(1./2)
+sin(2.*x))./(3+4.*x.^2).^2.*x).^4.*cos((3.^(1./2)+sin(2.*
x))./(3+4.*x.^2))+36.*tan(cos((3.^(1./2)+sin(2.*x))./(3+4.*x.^
2))).*(1+tan(cos((3.^(1./2)+sin(2.*x))./(3+4.*x.^2))).^2).*sin
((3.^(1./2)+sin(2.*x))./(3+4.*x.^2)).*(2.*cos(2.*x)./(3+4.*x.^
2)-8.*(3.^(1./2)+sin(2.*x))./(3+4.*x.^2).^2.*x).^2.*cos((3.^
(1./2)+sin(2.*x))./(3+4.*x.^2)).*(-4.*sin(2.*x)./(3+4.*x.^2)
-32.*cos(2.*x)./(3+4.*x.^2).^2.*x+128.*(3.^(1./2)+sin(2.*
x))./(3+4.*x.^2).^3.*x.^2-8.*(3.^(1./2)+sin(2.*x))./(3+4.*x.^
2).^2)+6.*tan(cos((3.^(1./2)+sin(2.*x))./(3+4.*x.^2))).*(1+tan
(cos((3.^(1./2)+sin(2.*x))./(3+4.*x.^2))).^2).*cos((3.^(1./2)+
sin(2.*x))./(3+4.*x.^2)).^2.*(2.*cos(2.*x)./(3+4.*x.^2)-8.*
(3.^(1./2)+sin(2.*x))./(3+4.*x.^2).^2.*x).^4+6.*tan(cos((3.^
(1./2)+sin(2.*x))./(3+4.*x.^2))).*(1+tan(cos((3.^(1./2)+sin(2.
*x))./(3+4.*x.^2))).^2).*sin((3.^(1./2)+sin(2.*x))./(3+4.*x.^
2)).^2.*(-4.*sin(2.*x)./(3+4.*x.^2)  32.*cos(2.*x)./(3+4.*x.^
2).^2.*x+128.*(3.^(1./2)+sin(2.*x))./(3+4.*x.^2).^3.*x.^2-8.*
(3.^(1./2)+sin(2.*x))./(3+4.*x.^2).^2).^2+8.*tan(cos((3.^(1./2)
+sin(2.*x))./(3+4.*x.^2))).*(1+tan(cos((3.^(1./2)+sin(2.*x))./
(3+4.*x.^2))).^2).*sin((3.^(1./2)+sin(2.*x))./(3+4.*x.^2)).^2.
*(2.*cos(2.*x)./(3+4.*x.^2)-8.*(3.^(1./2)+sin(2.*x))./(3+4.*
x.^2).^2.*x).*(-8.*cos(2.*x)./(3+4.*x.^2)+96.*sin(2.*x)./(3+
4.*x.^2).^2.*x+768.*cos(2.*x)./(3+4.*x.^2).^3.*x.^2-48.*cos(2.
*x)./(3+4.*x.^2).^2-3072.*(3.^(1./2)+sin(2.*x))./(3+4.*x.^2).^
4.*x.^3+384.*(3.^(1./2)+sin(2.*x))./(3+4.*x.^2).^3.*x)
      myxx=max(yxxxx), R=(h.^4).* abs(myxx./384)
```

将其保存为名为 myxx.m 的 M 文件,然后在 MATLAB 工作窗口输入该文件名

```
>> myxx
```

运行后屏幕显示 $myxx = \max_{-\pi\leqslant x\leqslant\pi} |f^{(4)}(x)|$ 和 $H_{n,3}(x)$ 在区间 $[-\pi,\pi]$ 上的误差公式 $R=\frac{h^4}{384}\max_{-\pi\leqslant x\leqslant\pi}|f^{(4)}(x)|$ 如下:

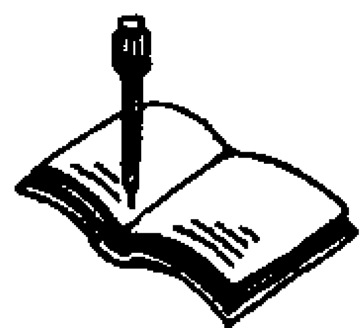
```
myxx =                          R =
        73.94706841647552   1734520780029061/9007199254740992*h^4
```

最后在 MATLAB 工作窗口输入

```
>>h=2*pi/9; R=1734520780029061/9007199254740992*h^4
```

运行后屏幕显示 $H_{n,3}(x)$ 在区间 $[-\pi,\pi]$ 上的误差限

```
R =
    0.04574453029948
```



习题 6.6

1. 作函数 $f(x)=\dfrac{1}{1+25x^2}$ 在区间 $[-5,5]$ 上的分段埃尔米特插值函数 $H_{n,3}(x)$ 的图形(按等距节点,分别取 $n=2,4,6,8,10$),并讨论分段埃尔米特插值函数 $H_{n,3}(x)$ 的次数与误差 $|R_n(x)|$ 的关系.

2. 设函数 $f(x)=\sin\left(\dfrac{2-\cos 4x}{1+25x^2}\right)$ 定义在区间 $[-\pi,\pi]$ 上,取 $n=13$,按等距节点构造分段埃尔米特插值函数 $H_{n,3}(x)$,并用 MATLAB 程序计算各小区间中点处 $H_{n,3}(x)$ 的值及其相对误差和绝对误差.

3. 设函数 $f(x)=\dfrac{1}{1+x^2}$ 定义在区间 $[-5,5]$ 上,取 $n=10$,按等距节点构造分段埃尔米特插值函数 $H_{n,3}(x)$,用 MATLAB 程序计算各小区间中点 x_i 处 $H_{n,3}(x)$ 的值,作出节点、插值点、$f(x)$ 和 $H_{n,3}(x)$ 的图形.

4. 设函数 $f(x)=0.15x^2-\sin(2x-1)$ 定义在区间 $[-\pi,\pi]$ 上,取 $n=7$,按等距节点构造分段埃尔米特插值函数 $H_{n,3}(x)$,用 MATLAB 程序计算各小区间中点 x_i 处 $H_{n,3}(x)$ 的值,作出节点、插值点、$f(x)$ 和 $H_{n,3}(x)$ 的图形.

5. 设函数 $f(x)=\tan\left(\cos\left(\dfrac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$ 定义在区间 $[-1,1]$ 上,取 $n=10$,按等距节点构造分段埃尔米特插值函数 $H_{n,3}(x)$.

(1) 用 MATLAB 程序计算各小区间中点 x_i 处 $H_{n,3}(x)$ 的值,作出节点、插值点、$f(x)$ 和 $H_{n,3}(x)$ 的图形;

(2) 用 MATLAB 程序计算各小区间中点处 $H_{n,3}(x)$ 的值及其绝对误差和相对误差;

(3) 用 MATLAB 程序估计 $\max\limits_{-\pi\leqslant x\leqslant\pi}|f^{(4)}(x)|$ 和 $H_{n,3}(x)$ 在区间 $[-\pi,\pi]$ 上的误差公式.

6.7 三次样条及其 MATLAB 程序

样条(spline)插值是一种既能克服高次多项式插值的缺陷,又能保证一定的光滑性的分段插值方法.样条的名称来源于船舶、飞机等设计中描绘光滑外形曲线用的绘图工具.一根有弹性的细长木条用压铁固定在节点上,其他地方让它自然弯曲,如此画出的曲线称为样条曲线(见图 6-17).从图 6-17 可见,这样

描绘出的样条曲线是将一段段的三次曲线拼成的曲线. 该曲线在拼接处不仅连续,而且具有二阶的连续导数,通常将这样得到的函数称作**样条函数**. 本节介绍几种样条插值函数,计算方法和误差估计及其 MATLAB 程序.

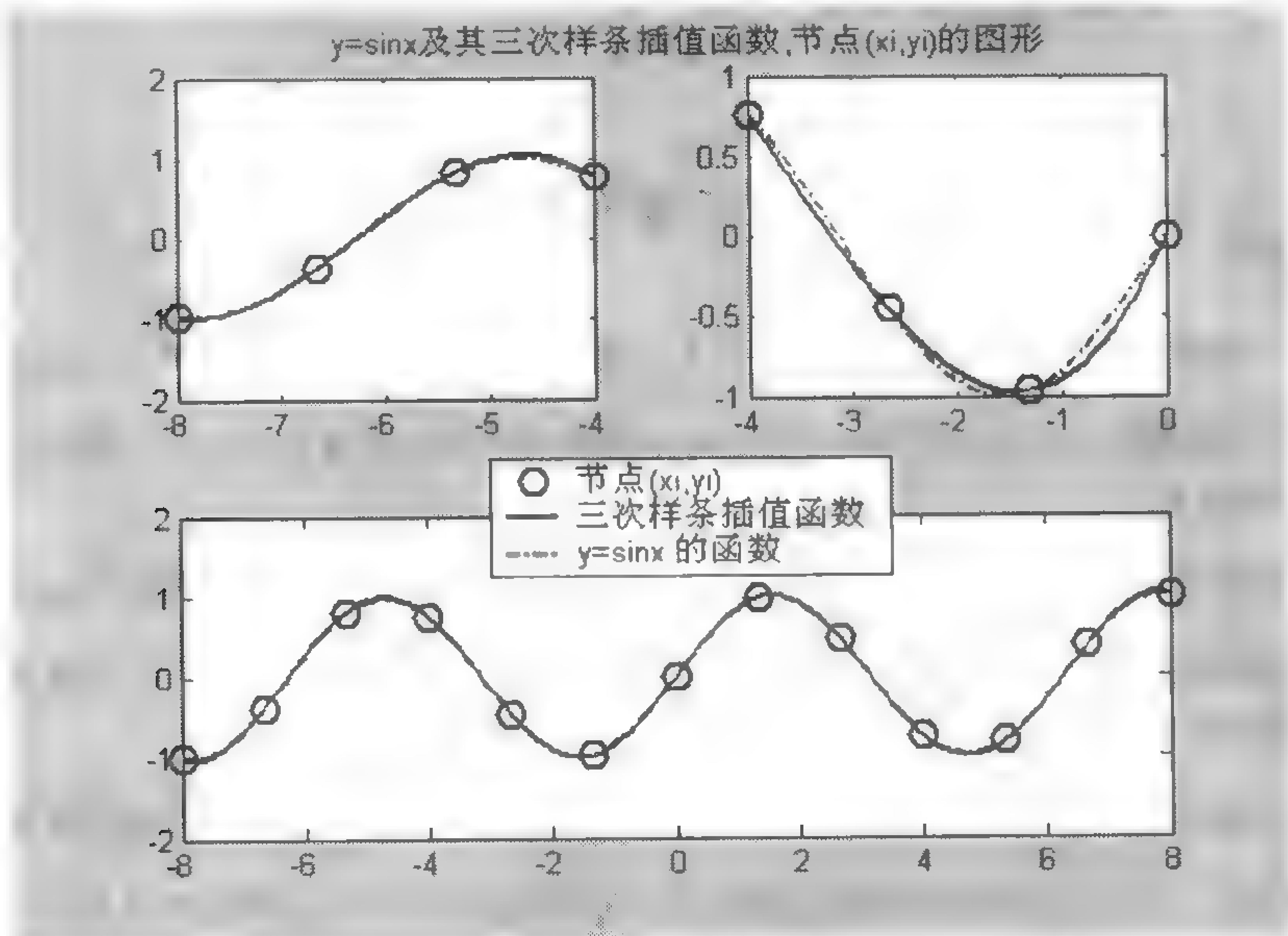


图 6-17 $y=\sin x$ 及其三次样条函数和节点的图形

6.7.1 三次样条函数

样条插值是分段插值. 因为样条曲线的曲率是处处连续的,所以要求样条函数的二阶导数连续. 人们普遍使用的样条函数是分段三次多项式. 我们可以用定义分段埃尔米特插值函数的方法,类似地给出样条插值函数的定义.

定义 6.5 设函数 $f(x)$ 在 $[a,b]$ 上的 $n+1$ 个点 $a=x_0<x_1<x_2<\cdots<x_n=b$ 处的函数值为 $f(x_i)=y_i(i=0,1,2,\cdots,n)$. 连接每两个相邻的点 (x_i,y_i) 和 (x_{i+1},y_{i+1}),作一条曲线函数 $S_n(x)$,使得 $S_n(x)$ 满足如下条件:

(1) $S_n(x)$ 在 $[a,b]$ 上具有连续的二阶导数;

(2) $S_n(x_i)=y_i(i=0,1,2,\cdots,n)$; (6.59)

(3) $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上是三次多项式 $s_i(x)$ $(i=1,2,\cdots,n)$,

则称曲线函数 $S_n(x)$ 为 $f(x)$ 在点 (x_i,y_i) $(i=0,1,2,\cdots,n)$ 处的三次样条函数.

定义 6.5 中条件(1)的几何意义是:在 $[a,b]$ 上一阶导数 $S_n'(x)$ 连续意味着曲线 $S_n(x)$ 没有急弯,二阶导数 $S_n''(x)$ 连续意味着曲线 $S_n(x)$ 在每一点的曲率半径有定义(见图 6-17). 图 6-17 是在区间 $[-8,8]$,$[-8,-4]$ 和 $[-4,0]$ 上被插值函数 $y=\sin x$ 及其在节点 (x_i,y_i) (其中 $x_i=-8+\frac{4i}{3}$ $(i=0,1,2,\cdots,12)$)

处的三次样条函数和节点的图形. 图中小圆圈表示节点(x_i,y_i),实线表示三次样条函数,虚线表示被插值函数 $y=\sin x$ 的曲线. 在节点附近三次样条函数的曲线不但光滑,而且与被插值函数的误差小,几乎与 $y=\sin x$ 的曲线重合.

虽然分段三次埃尔米特插值函数和三次样条函数都是分段三次插值函数,但是由于三次样条函数 $S_n(x)$ 在$[a,b]$上具有连续的二阶导数,而三次埃尔米特插值函数 $H_{n,3}(x)$ 在$[a,b]$上只具有连续的一阶导数,所以在曲线的凹凸性变化比较大的某些局部区域上三次埃尔米特插值函数与被插值函数的误差比三次样条函数的大(见图 6-18). 但是在相对较长的某些局部区域上,如果曲线的凹凸性不变,且非常平缓,则有时三次埃尔米特插值函数与被插值函数的误差比三次样条函数的小(见图 6-19 中在$[2,\pi]$ 上的图形). 图 6-18 和图 6-19 分别是被插值函数 $y=\cos x$ 和 $f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$ 在节点(x_i,y_i)处的三次样条函数、三次埃尔米特插值函数和节点的图形. 图 6-18 中小圆圈表示节点(x_i,y_i)(其中 $x_i=-8+\frac{4i}{3}, i=0,1,2,\cdots,12$),实线表示三次样条函数,虚线表示三次埃尔米特插值函数,而点划线表示被插值函数 $y=\cos x$ 的曲线. 由图可见,总体上来说,分段三次埃尔米特插值函数与被插值函数的误差比三次样条函数的大,并且三次样条函数的曲线几乎与被插值函数的曲线重合.

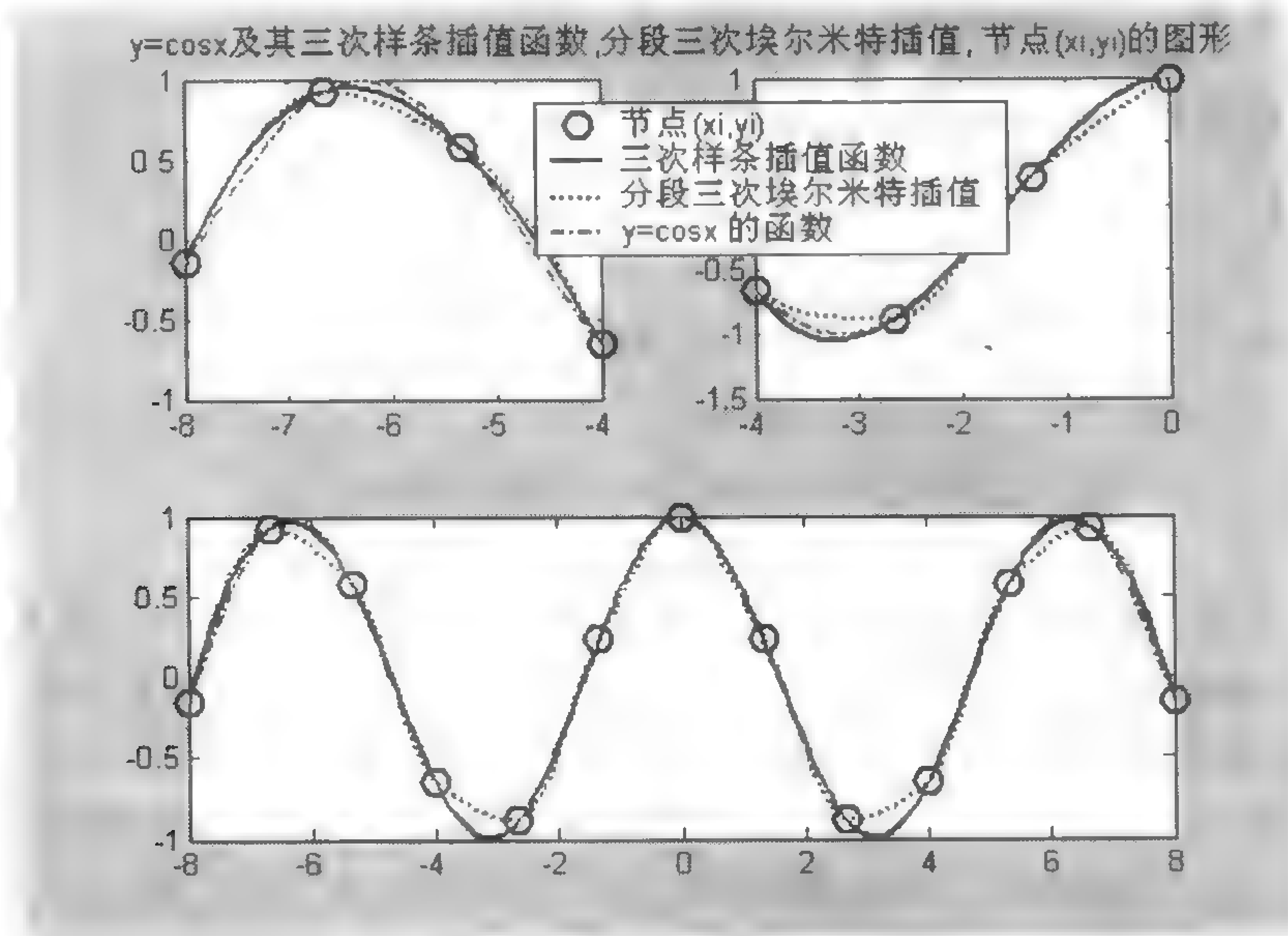


图 6-18　函数 $y=\cos x$ 及其三次样条插值函数、分段三次埃尔米特插值函数和节点的图形

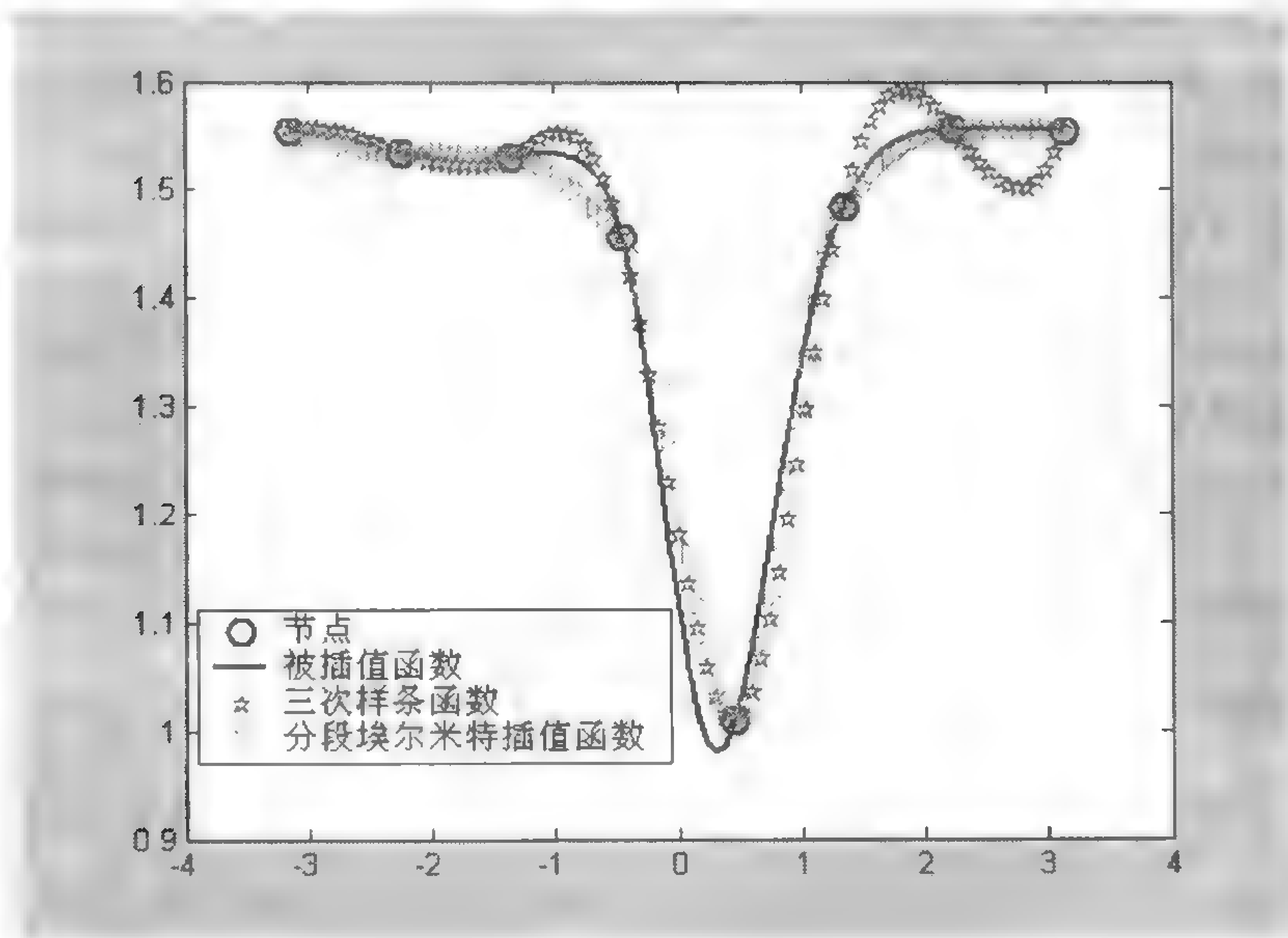


图 6-19 函数 $f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$

6.7.2 三次样条函数的存在性

由定义 6.5 中的条件(3),不妨将 $S_n(x)$ 记为

$$S_n(x)=\{s_i(x)\mid s_i(x)=a_ix^3+b_ix^2+c_ix+d_i,x\in[x_{i-1},x_i],i=1,2,\cdots,n\},\tag{6.60}$$

其中 a_i,b_i,c_i,d_i 为待定系数,共 $4n$ 个. 条件(1)用数学式子可以表示为

$$\begin{cases}s_i(x_i)=s_{i+1}(x_i),\\ s_i'(x_i)=s_{i+1}'(x_i), & i=1,2,\cdots,n-1.\\ s_i''(x_i)=s_{i+1}''(x_i),\end{cases}\tag{6.61}$$

容易看出,(6.59),(6.61)式共含有 $4n-2$ 个方程. 为确定 $S_n(x)$ 的 $4n$ 个待定参数,尚需再给出 2 个条件. 为确定 $S_n(x)$ 在各个子区间 $[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$ 上三次多项式 $s_i(x)$ 的表达式,自然会联想到前节的分段埃尔米特插值,因为在每个子区间 $[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$ 上分段三次埃尔米特插值函数 $H_{n,3}(x)$ 也是三次多项式,但是需要 $f'(x_i)=y_i'(i=0,1,2,\cdots,n)$ 为已知. 如果能利用三次样条函数 $S_n(x)$ 在 $[a,b]$ 上具有连续的二阶导数的条件确定 $f'(x_i)$,那么由(6.52)式,三次样条函数 $S_n(x)$ 便可唯一的表示出来. 为此设 $f'(x_i)=y'_i(i$

$=0,1,2,\cdots,n$），$h_{i-1}=x_i-x_{i-1}$（$i=1,2,\cdots,n$），则 $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上的三次多项式 $s_i(x)$ （$i=1,2,\cdots,n$）为

$$
\begin{aligned}
S_n(x)=s_i(x)=&\frac{y_{i-1}}{h_{i-1}^3}(h_{i-1}-2x_{i-1}+2x)(x-x_i)^2+\\
&\frac{y_i}{h_{i-1}^3}(h_{i-1}+2x_i-2x)(x-x_{i-1})^2+\\
&\frac{y'_{i-1}}{h_{i-1}^2}(x-x_{i-1})(x-x_i)^2+\frac{y'_i}{h_{i-1}^2}(x-x_{i-1})^2(x-x_i).
\end{aligned}
$$

上式对 x 求二阶导函数为

$$
\begin{aligned}
S''_n(x)=s''_i(x)=&\frac{6y_{i-1}}{h_{i-1}^3}(h_{i-1}-2x_i+2x)+\frac{6y_i}{h_{i-1}^3}(h_{i-1}+2x_{i-1}-2x)+\\
&\frac{2y'_{i-1}}{h_{i-1}^2}(h_{i-1}-3x_i+3x)+\frac{2y'_i}{h_{i-1}^2}(-h_{i-1}-3x_{i-1}+3x),
\end{aligned}
$$

所以，$S''_n(x)$ 在点 x_i 处的左极限为

$$S''_n(x_i-0)=s''_i(x_i)=\frac{6y_{i-1}}{h_{i-1}^2}-\frac{6y_i}{h_{i-1}^2}+\frac{2y'_{i-1}}{h_{i-1}}+\frac{4y'_i}{h_{i-1}}. \tag{6.62}$$

$S''_n(x)$ 在点 x_i 处的右极限为

$$S''_n(x_i+0)=s''_{i+1}(x_i)=-\frac{6y_i}{h_i^2}+\frac{6y_{i+1}}{h_i^2}-\frac{4y'_i}{h_i}-\frac{2y'_{i+1}}{h_i} \tag{6.63}$$

因为 $S''_n(x)$ 在点 x_i 处连续，即 $S''_n(x_i-0)=S''_n(x_i+0)$，所以

$$\frac{6y_{i-1}}{h_{i-1}^2}-\frac{6y_i}{h_{i-1}^2}+\frac{2y'_{i-1}}{h_{i-1}}+\frac{4y'_i}{h_{i-1}}=-\frac{6y_i}{h_i^2}+\frac{6y_{i+1}}{h_i^2}-\frac{4y'_i}{h_i}-\frac{2y'_{i+1}}{h_i}. \tag{6.64}$$

记
$$\lambda_i=\frac{h_{i-1}}{h_i+h_{i-1}},\quad \mu_i=1-\lambda_i=\frac{h_i}{h_i+h_{i-1}},$$

$$d_i=3\left[\frac{\mu_i}{h_{i-1}}(y_i-y_{i-1})+\frac{\lambda_i}{h_i}(y_{i+1}-y_i)\right]\ (i=1,2,\cdots,n-1).$$

将(6.64)式整理得方程组

$$\mu_i y'_{i-1}+2y'_i+\lambda_i y'_{i+1}=d_i\quad (i=1,2,\cdots,n-1).$$

可以得到下面的定理.

定理 6.11 如果 $S_n(x)$ 是定义 6.5 所定义的 $y=f(x)$ 在点 (x_i,y_i) （$i=0,1,2,\cdots,n$）处的三次样条函数，则

(1) $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上的三次多项式 $s_i(x)$ （$i=1,2,\cdots,n$）为

$$
\begin{aligned}
S_n(x)=s_i(x)=&\frac{y_{i-1}}{h_{i-1}^3}(h_{i-1}-2x_{i-1}+2x)(x-x_i)^2+\\
&\frac{y_i}{h_{i-1}^3}(h_{i-1}+2x_i-2x)(x-x_{i-1})^2+
\end{aligned}
$$

$$\frac{y'_{i-1}}{h^2_{i-1}}(x-x_{i-1})(x-x_i)^2+\frac{y'_i}{h^2_{i-1}}(x-x_{i-1})^2(x-x_i), \tag{6.65}$$

其中 y'_{i-1} 和 $y'_i(i=1,2,\cdots,n)$ 是线性方程组

$$\mu_i y'_{i-1}+2y'_i+\lambda_i y'_{i+1}=d_i \quad (i=1,2,\cdots,n-1) \tag{6.66}$$

的一组解. 其中 $y_i=f(x_i)$, $y'_i=S'_n(x_i)$ $(i=0,1,2,\cdots,n)$, $h_{i-1}=x_i-x_{i-1}$, $\lambda_i=\dfrac{h_{i-1}}{h_{i-1}+h_i}$, $\mu_i=1-\lambda_i=\dfrac{h_i}{h_{i-1}+h_i}$, $d_i=3\left[\dfrac{\mu_i}{h_{i-1}}(y_i-y_{i-1})+\dfrac{\lambda_i}{h_i}(y_{i+1}-y_i)\right]$ $(i=1,2,\cdots,n-1)$.

(2) $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上的二阶导函数为

$$\begin{aligned}S''_n(x)=s''_i(x)=&\frac{6y_{i-1}}{h^3_{i-1}}(h_{i-1}-2x_i+2x)+\frac{6y_i}{h^3_{i-1}}(h_{i-1}+2x_{i-1}-2x)+\\&\frac{2y'_{i-1}}{h^2_{i-1}}(h_{i-1}-3x_i+3x)+\frac{2y'_i}{h^2_{i-1}}(-h_{i-1}-3x_{i-1}+3x).\end{aligned} \tag{6.67}$$

(3) $S_n(x)$ 在区间 $[x_0,x_n]$ 端点 x_0 和 x_n 处的二阶导函数为

$$S''_n(x_0)=\frac{6(y_1-y_0)}{h^2_0}-\frac{2(2y'_0+y'_1)}{h_0}, \tag{6.68}$$

$$S''_n(x_n)=\frac{6(y_{n-1}-y_n)}{h^2_{n-1}}+\frac{2(y'_{n-1}+2y'_n)}{h_{n-1}}. \tag{6.69}$$

(4) $S_n(x)$ 在区间 $[x_0,x_n]$ 端点 x_0 和 x_n 处的三阶导函数为

$$S'''_n(x_0)=\frac{12(y_0-y_1)}{h^3_0}+\frac{6(y'_1+y'_0)}{h^2_0}, \tag{6.70}$$

$$S'''_n(x_n)=\frac{12(y_{n-1}-y_n)}{h^3_{n-1}}+\frac{6(y'_n+y'_{n-1})}{h^2_{n-1}}. \tag{6.71}$$

我们还可以证明,(6.66)式还可以用 $S_n(x)$ 在区间端点 x_{i-1}, x_i 和 x_{i+1} 处的二阶导数表示,从而得到下面推论.

推论 6.3 如果 $S_n(x)$ 是定义 6.5 定义的 $f(x)$ 在点 (x_i,y_i) $(i=0,1,2,\cdots,n)$ 处的三次样条函数,则

(1) $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上的三次多项式 $s_i(x)$ $(i=1,2,\cdots,n)$ 还可以表示为

$$\begin{aligned}S_n(x)=s_i(x)=&\frac{S''_n(x_{i-1})}{6h_{i-1}}(x_i-x)^3+\frac{S''_n(x_i)}{6h_{i-1}}(x-x_{i-1})^3+\\&p_{i-1}(x_i-x)+q_{i-1}(x-x_{i-1}),\end{aligned} \tag{6.72}$$

其中

$$p_i=\frac{y_{i-1}}{h_{i-1}}-\frac{S_n''(x_{i-1})}{6}h_{i-1},\ q_i=\frac{y_i}{h_{i-1}}-\frac{S_n''(x_i)}{6}h_{i-1},$$

而 $S_n''(x_i)$ 和 $S_n''(x_{i-1})$ $(i=1,2,\cdots,n)$ 是线性方程组

$$h_{i-1}S_n''(x_{i-1})+2(h_{i-1}+h_i)S_n''(x_i)+h_iS_n''(x_{i+1})$$
$$=6(b_i-b_{i-1})\ (i=1,2,\cdots,n-1) \tag{6.73}$$

的一组解,其中 $b_i=\frac{y_{i+1}-y_i}{h_i}$.

(2) $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ $(i=1,\cdots,n-1,n)$ 上的一阶导函数为

$$S_n'(x)=-\frac{S_n''(x_{i-1})}{2h_{i-1}}(x_i-x)^2+\frac{S_n''(x_i)}{2h_{i-1}}(x_{i-1}-x)^2$$
$$+\frac{y_i-y_{i-1}}{h_{i-1}}+\frac{S_n''(x_{i-1})-S_n''(x_i)}{6}h_{i-1}. \tag{6.74}$$

证明 (1) 根据(6.67)式,可得到 $S_n(x)$ 在区间 $[x_{i-1},x_i]$ 和 $[x_i,x_{i+1}]$ 的端点 x_{i-1},x_i 和 x_{i+1} 处的二阶导数

$$h_{i-1}S_n''(x_{i-1})=\frac{6(y_i-y_{i-1})}{h_{i-1}}-4y_{i-1}'-2y_i', \tag{6.75}$$

$$h_{i-1}S_n''(x_i)=-\frac{6(y_i-y_{i-1})}{h_{i-1}}+2y_{i-1}'+4y_i', \tag{6.76}$$

$$h_iS_n''(x_i)=-\frac{6(y_i-y_{i+1})}{h_i}-2y_{i+1}'-4y_i', \tag{6.77}$$

$$h_iS_n''(x_{i+1})=\frac{6(y_i-y_{i+1})}{h_i}+4y_{i+1}'+2y_i'. \tag{6.78}$$

(6.76) ×2 + (6.75) 得

$$h_{i-1}S_n''(x_{i-1})+2h_{i-1}S_n''(x_i)=-\frac{6(y_i-y_{i-1})}{h_{i-1}}+6y_i', \tag{6.79}$$

(6.77) ×2 + (6.78) 得

$$h_iS_n''(x_{i+1})+2h_iS_n''(x_i)=\frac{6(y_{i+1}-y_i)}{h_i}-6y_i'. \tag{6.80}$$

(6.79)与(6.80)之和为(6.73)式. 由(6.80)解得 y_i',代入(6.75),解得 y_{i-1}',再代入(6.65),化简整理得(6.72).

(2) 将(6.72)式两边对 x 求导,整理得(6.74)式.

6.7.3 端点约束条件

定理6.11 的(6.65),(6.66)和推论6.3 的(6.72),(6.73)分别提供了计算三次样条函数(6.60)的两种计算方法. (6.66)式是关于 $n+1$ 个未知量 y'_k $(k=0,1,\cdots,n)$ 的 $n-1$ 个线性方程组,有无穷多组解. 而(6.73)式也是关于 $n+1$ 个

未知量 $S''_n(x_k)$ $(k=0,1,\cdots,n)$ 的 $n-1$ 个线性方程组，有无穷多组解. 在实际问题中，往往需要根据具体情况补充两个附加条件，通常称为**端点约束条件**，用来唯一确定线性方程组(6.60)的一组解，即 $S(x)$ 被唯一确定的条件. 常见的端点约束条件如表 6-7 所示.

表 6-7 端点约束条件

序号	三次样条的种类	一阶导数 $f'(x_0)$ 和 $f'(x_n)$	端点约束条件
1	压紧样条(clamped spline)(已知一阶导数，这是最佳选择)	已知	$S'_n(x_0)=f'(x_0),S'_n(x_n)=f'(x_n)$
2	周期端点样条(当端点值 $f(x_0)=f(x_n)$)	未知	$S_n(x_0)=S_n(x_n),S'_n(x_0)=S'_n(x_n)$, $S''_n(x_0)=S''_n(x_n)$
3	自然样条(natural spline)(常用的方法)	未知	$S''_n(x_0)=0$, $S''_n(x_n)=0$
4	外推样条(extrapolated spline)(由在 x_1,x_2 处的二阶导数外推到 $S''_n(x_0)$；由在 x_{n-1},x_{n-2} 处的二阶导数外推到 $S''_n(x_n)$)	未知	$S''_n(x_0)=S''_n(x_1)-\dfrac{h_0}{h_1}[S''_n(x_2)-S''_n(x_1)]$ $S''_n(x_n)=S''_n(x_{n-1})-\dfrac{h_{n-1}}{h_{n-2}}[S''_n(x_{n-1})-S''_n(x_{n-2})]$
5	抛物线结尾样条(parabolically terminated Spline)	未知	对于 $\forall x\in[x_0,x_1]$，有 $S'''_n(x)\equiv 0$ 对于 $\forall x\in[x_{n-1},x_n]$，有 $S'''_n(x)\equiv 0$ 例如，取 $S''_n(x_0)=S''_n(x_1),S''_n(x_n)=S''_n(x_{n-1})$
6	端点调整曲率样条(End-point Curvature-adjusted Spline)	未知	给定 $S''_n(x_0)=m_0,S''_n(x_n)=m_n$, 例如，取 $S''_n(x_0)=S''_n(x_1),S''_n(x_n)=S''_n(x_{n-1})$

6.7.4 用一阶导数计算的几种样条函数

我们可以证明，在给定了表 6-7 中六种端点约束条件之一时，则定理 6.11 的(6.65)、(6.66)和推论 6.3 的(6.72)、(6.73)分别有唯一解，由插值多项式的唯一性，故 $S_n(x)$ 被唯一确定. 同时也推导出分别用一阶导数或二阶导数计算

这几种样条函数的两类计算公式.

定理 6.12(端点调整曲率样条) (1) 如果定义三次样条函数 $S_n(x)$ 在区间端点的二阶导数 $S_n''(x_0)=m_0$ 和 $S_n''(x_n)=m_n$,则存在唯一的三次样条函数

$$S_n(x)=\{s_i(x),\ x\in[x_{i-1},x_i],\ i=1,2,\cdots,n\},$$

且 $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上的三次多项式 $s_i(x)$ $(i=1,2,\cdots,n)$ 为 (6.65) 式,即

$$\begin{aligned}S_n(x)=s_i(x)=&\frac{y_{i-1}}{h_{i-1}^3}(h_{i-1}-2x_{i-1}+2x)(x-x_i)^2+\\&\frac{y_i}{h_{i-1}^3}(h_{i-1}+2x_i-2x)(x-x_{i-1})^2+\\&\frac{y'_{i-1}}{h_{i-1}^2}(x-x_{i-1})(x-x_i)^2+\frac{y'_i}{h_{i-1}^2}(x-x_{i-1})^2(x-x_i),\end{aligned}$$

其中 $S_n'(x_{i-1})=y'_{i-1},S_n'(x_i)=y'_i$ $(i=0,1,\cdots,n)$ 是三对角方程组

$$\begin{pmatrix}2 & \lambda_0 & & & & \\ \mu & 2 & \lambda_1 & & & \\ & \mu_2 & 2 & \lambda_2 & & \\ & & \mu_3 & 2 & \ddots & \\ & & & \ddots & \ddots & \lambda_{n-1} \\ & & & & \mu_n & 2\end{pmatrix}\begin{pmatrix}y'_0\\ y'_1\\ y'_2\\ \vdots\\ y'_{n-1}\\ y'_n\end{pmatrix}=\begin{pmatrix}d_0\\ d_1\\ d_2\\ \vdots\\ d_{n-1}\\ d_n\end{pmatrix}\tag{6.81}$$

的解,且(6.81)有且只有唯一解. 其中

$$h_{i-1}=x_i-x_{i-1}\ (i=1,2,\cdots,n),\ \lambda_i=\frac{h_{i-1}}{h_{i-1}+h_i},\ \mu_i=1-\lambda_i=\frac{h_i}{h_{i-1}+h_i},$$

$$d_i=3\left[\frac{\mu_i}{h_{i-1}}(y_i-y_{i-1})+\frac{\lambda_i}{h_i}(y_{i+1}-y_i)\right]\ (i=1,2,\cdots,n-1),\tag{6.82}$$

$$\lambda_0=1,\mu_n=1,d_0=\frac{3}{h_0}(y_1-y_0)-\frac{m_0h_0}{2},d_n=\frac{3}{h_{n-1}}(y_n-y_{n-1})+\frac{m_nh_{n-1}}{2}.\tag{6.83}$$

(2) 当取 $m_0=S_n''(x_0)=S_n''(x_1),m_n=S_n''(x_n)=S_n''(x_{n-1})$ 时,则(6.81)的系数是(6.82)和

$$\lambda_0=2,\mu_n=2,d_0=\frac{4}{h_0}(y_1-y_0),d_n=\frac{4}{h_{n-1}}(y_n-y_{n-1}).\tag{6.84}$$

证明 (1) 由 $S_n''(x_0)=m_0,S_n''(x_n)=m_n$ 和(6.68),(6.69),(6.66)式,得

$$\begin{cases}2y'_0+y'_1=\dfrac{3}{h_0}(y_1-y_0)-\dfrac{m_0h_0}{2},\\ \mu_iy'_{i-1}+2y'_i+\lambda_iy'_{i+1}=d_i\ \ (i=1,2,\cdots,n-1),\\ y'_{n-1}+2y'_n=\dfrac{3}{h_{n-1}}(y_n-y_{n-1})+\dfrac{m_nh_{n-1}}{2}.\end{cases}\tag{6.85}$$

把(6.85)式写成矩阵方程(6.81),其中系数为(6.82)和(6.83). 因为(6.81)式的系数矩阵具有严格对角占优的优势,即主对角线上的元素的绝对值大于该行的其他元素的绝对值之和,所以(6.81)式的系数矩阵是非奇异矩阵.因此矩阵方程(6.81)有且只有唯一解.

(2) 当取 $m_0 = S_n''(x_0) = S_n''(x_1)$, $m_n = S_n''(x_n) = S_n''(x_{n-1})$ 时,由(6.62)和(6.63)式,得

$$m_0 = S_n''(x_0) = s_1''(x_0) = -\frac{6y_0}{h_0^2} + \frac{6y_1}{h_0^2} - \frac{4y_0'}{h_0} - \frac{2y_1'}{h_0},$$

$$m_0 = S_n''(x_1) = s_1''(x_1) = \frac{6y_0}{h_0^2} - \frac{6y_1}{h_0^2} + \frac{2y_0'}{h_0} + \frac{4y_1'}{h_0},$$

$$m_n = S_n''(x_{n-1}) = s_n''(x_{n-1}) = -\frac{6y_{n-1}}{h_{n-1}^2} + \frac{6y_n}{h_{n-1}^2} - \frac{4y_{n-1}'}{h_{n-1}} - \frac{2y_n'}{h_{n-1}},$$

$$m_n = S_n''(x_n) = s_n''(x_n) = \frac{6y_{n-1}}{h_{n-1}^2} - \frac{6y_n}{h_{n-1}^2} + \frac{2y_{n-1}'}{h_{n-1}} + \frac{4y_n'}{h_{n-1}},$$

代入(6.85)式,解得,

$$\begin{cases} 2y_0' + 2y_1' = \dfrac{4}{h_0}(y_1 - y_0), \\ \mu_i y_{i-1}' + 2y_i' + \lambda_i y_{i+1}' = d_i \quad (i = 1, 2, \cdots, n-1), \\ 2y_{n-1}' + 2y_n' = \dfrac{4}{h_{n-1}}(y_n - y_{n-1}). \end{cases} \tag{6.86}$$

把(6.86)式写成矩阵方程(6.81),其中系数为(6.82)和(6.84). 因为(6.81)式的系数矩阵具有严格对角占优的优势,即主对角线上的元的绝对值大于该行的其他元的绝对值之和,所以(6.81)式的系数矩阵是非奇异的,因此矩阵方程(6.81)有且只有唯一解.

注意 从几何角度分析,通过对 $S_n''(x_0) = m_0$, $S_n''(x_n) = m_n$ 赋值,可以调整区间端点的曲率,使得 $S_n(x)$ 与 $f(x)$ 的误差更小.

推论 6.4(抛物线结尾样条) 如果取被插值函数 $y = f(x)$ 的端点约束条件为对于任意 $x \in [x_0, x_1]$,有 $S_n'''(x) \equiv 0$,且对于任意 $x \in [x_{n-1}, x_n]$,有 $S_n'''(x) \equiv 0$,则存在唯一的 $f(x)$ 的三次样条函数 $S_n(x)$,且 $S_n(x)$ 在每个子区间 $[x_{i-1}, x_i]$ 上的三次多项式 $s_i(x)$ $(i = 1, 2, \cdots, n)$ 为(6.65)式,其中 $S_n'(x_{i-1}) = y_{i-1}'$, $S_n'(x_i) = y_i'$ $(i = 1, 2, \cdots, n)$ 是三对角方程组(6.81)的解,且(6.81)有且只有唯一解. 其中矩阵方程(6.81)的系数为(6.82)和(6.84).

证明 由 $S_n'''(x_0) = S_n'''(x_1) = 0$ 和 $S_n'''(x_{n-1}) = S_n'''(x_n) = 0$,得

$$m_0 = S_n''(x_0) = S_n''(x_1) = c \text{ 和 } m_n = S_n''(x_{n-1}) = S_n''(x_n) = c.$$

由定理 6.12 的(2)得到结论.

注意 端点约束条件为对于任意 $x\in[x_0,x_1]$，有 $S'''_n(x)\equiv 0$，且对于任意 $x\in[x_{n-1},x_n]$，有 $S'''_n(x)\equiv 0$，使得三次多项式 $S_n(x)$ 退化为二次多项式(见例 6.7.5).

推论 6.5(自然样条) 如果已知被插值函数 $y=f(x)$ 在两端点的二阶导数 $f''(x_0)=f''(x_n)=0$，取端点约束条件为 $S''_n(x_0)=S''_n(x_n)=0$，则存在唯一的三次样条函数 $S_n(x)$，且 $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上的三次多项式 $s_i(x)$ $(i=1,2,\cdots,n)$ 为(6.65)式，其中 $S'_n(x_{i-1})=y'_{i-1}, S'_n(x_i)=y'_i$ $(i=1,2,\cdots,n)$ 是三对角方程组(6.81)的解，且(6.81)有且只有唯一解. 其中矩阵方程(6.81)的系数为(6.82)和

$$\lambda_0=1,\ \mu_n=1,\ d_0=\frac{3}{h_0}(y_1-y_0),\ d_n=\frac{3}{h_{n-1}}(y_n-y_{n-1}). \tag{6.87}$$

证明 因为 $S''_n(x_0)=m_0=0, S''_n(x_n)=m_n=0$，根据定理 6.12 的(1)中(6.85)式为

$$\begin{cases}2y'_0+y'_1=\dfrac{3}{h_0}(y_1-y_0),\\ \mu_i y'_{i-1}+2y'_i+\lambda_i y'_{i+1}=d_i\quad(i=1,2,\cdots,n-1),\\ y'_{n-1}+2y'_n=\dfrac{3}{h_{n-1}}(y_n-y_{n-1}).\end{cases} \tag{6.88}$$

把(6.88)式写成矩阵方程(6.81)，其中系数为(6.82)和(6.87). 因为(6.81)式的系数矩阵是严格对角占优，所以(6.81)式的系数矩阵是非奇异的，因此矩阵方程(6.81)有且只有唯一解.

注意 自然样条是柔软而有弹性的细长木条经过所有的节点，自然弯曲所画出的曲线，但让端点的斜率自由地在某一位置保持平衡，使得曲线的摇摆程度为最小. 它对有多位有效数字精度的试验数据进行曲线拟合时很有用.

推论 6.6(压紧样条) 如果已知被插值函数 $y=f(x)$ 的一阶导数 $y'_0=f'(x_0)$ 和 $y'_n=f'(x_n)$，取端点约束条件为 $S'_n(x_0)=y'_0, S'_n(x_n)=y'_n$，存在唯一的三次样条函数 $S_n(x)$，且 $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上的三次多项式 $s_i(x)$ $(i=1,2,\cdots,n)$ 为(6.65)式，其中 $S'_n(x_{i-1})=y'_{i-1}, S'_n(x_i)=y'_i$ $(i=1,2,\cdots,n)$ 是三对角方程组(6.81)的解，且(6.81)有且只有唯一解. 其中矩阵方程(6.81)的系数为(6.82)和

$$\lambda_0=0,\ \mu_n=0,\ d_0=2y'_0,\ d_n=2y'_n. \tag{6.89}$$

证明 由 $S'_n(x_0)=y'_0, S'_n(x_n)=y'_n$ 和(6.66)式，得

$$\begin{cases}2y'_0=2y'_0,\\ \mu_i y'_{i-1}+2y'_i+\lambda_i y'_{i+1}=d_i\quad(i=1,2,\cdots,n-1),\\ 2y'_n=2y'_n.\end{cases} \tag{6.90}$$

把(6.90)式写成矩阵方程(6.81),其中系数为(6.82)和(6.89).因为(6.81)式的系数矩阵是严格对角占优,所以(6.81)式的系数矩阵是非奇异的.因此矩阵方程(6.81)有且只有唯一解.

注意 因为压紧样条在端点有斜率.从几何角度分析,当柔软而有弹性的细长木条受外力使其经过所有的节点时所画出的曲线是压紧样条.它对需要绘制经过多个点的光滑曲线的绘图员提供帮助.

例 6.7.1 观测得函数 $y=f(x)$ 在若干点处的值为 $f(0)=0, f(2)=16, f(4)=36, f(6)=54, f(10)=82$ 和 $f'(0)=8, f'(10)=7$. 试求 $f(x)$ 的三次样条函数,并计算 $f(3)$ 和 $f(8)$ 的近似值.

解 (1) 令 $x_0=0, x_1=2, x_2=4, x_3=6, x_4=10, y_0=0, y_1=16, y_2=36, y_3=54, y_4=82$ 和 $y_0'=8, y_4'=7$.

根据题意知所要求的三次样条函数为压紧样条,根据推论 6.6 计算

$$h_0=h_1=h_2=2, h_3=4, \lambda_0=0, \lambda_1=\frac{h_0}{h_0+h_1}=\frac{1}{2},$$

$$\lambda_2=\frac{h_1}{h_1+h_2}=\frac{1}{2}, \lambda_3=\frac{h_2}{h_2+h_3}=\frac{1}{3},$$

$$\mu_1=1-\lambda_1=\frac{1}{2}, \mu_2=1-\lambda_2=\frac{1}{2}, \mu_3=1-\lambda_3=\frac{2}{3}, \mu_4=0, d_0=2y_0'=16,$$

$$d_1=3\left[\frac{\mu_1}{h_0}(y_1-y_0)+\frac{\lambda_1}{h_1}(y_2-y_1)\right]=3\left[\frac{1}{4}(16-0)+\frac{1}{4}(36-16)\right]=27,$$

$$d_2=3\left[\frac{\mu_2}{h_1}(y_2-y_1)+\frac{\lambda_2}{h_2}(y_3-y_2)\right]=3\left[\frac{1}{4}(36-16)+\frac{1}{4}(54-36)\right]=\frac{57}{2},$$

$$d_3=3\left[\frac{\mu_3}{h_2}(y_3-y_2)+\frac{\lambda_3}{h_3}(y_4-y_3)\right]=3\left[\frac{1}{3}(54-36)+\frac{1}{12}(82-54)\right]=25,$$

$$d_4=2y_4'=14.$$

根据(6.90)式,得矩阵方程

$$\begin{pmatrix} 2 & 0 & & & \\ \frac{1}{2} & 2 & \frac{1}{2} & & \\ & \frac{1}{2} & 2 & \frac{1}{2} & \\ & & \frac{1}{2} & 2 & \frac{1}{3} \\ & & & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} y_0' \\ y_1' \\ y_2' \\ y_3' \\ y_4' \end{pmatrix} = \begin{pmatrix} 16 \\ 27 \\ \frac{57}{2} \\ 25 \\ 14 \end{pmatrix}$$

解得 $y_1'=9, y_2'=10, y_3'=8$. 代入(6.65)计算样条的系数可得三次样条为

$$s_1(x)=4x^2(3-x)+2x(x-2)^2+\frac{9}{4}x^2(x-2), \quad 0\leqslant x\leqslant 2,$$

$$s_2(x)=4(x-1)(x-4)^2+9(5-x)(x-2)^2+\frac{9}{4}(x-2)(x-4)^2+\frac{5}{2}(x-4)(x-2)^2,\quad 2\leqslant x\leqslant 4,$$

$$s_3(x)=9(-3+x)(x-6)^2+\frac{27}{2}(7-x)(x-4)^2+\frac{5}{2}(x-4)(x-6)^2+2(x-6)(x-4)^2,\quad 4\leqslant x\leqslant 6,$$

$$s_4(x)=\frac{27}{16}(x-4)(x-10)^2+\frac{41}{16}(12-x)(x-6)^2+\frac{1}{2}(x-6)(x-10)^2+\frac{7}{16}(x-10)(x-6)^2,6\leqslant x\leqslant 10.$$

计算得 $f(3)\approx s_2(3)=25.75$ 和 $f(8)\approx s_4(8)=68.5$.

从例 6.7.1 可以看出,用手工计算三次样条函数很复杂,根据推论 6.6 现编写下面的计算压紧样条函数及其 $h_{i-1}(i=1,2,\cdots,n)$, $\lambda_i,\mu_i,d_i(i=1,2,\cdots,n-1)$ 和线性方程组(6.90)的解的 MATLAB 程序,读者使用时,只需输入节点 (x_i,y_i) $(i=0,1,2,\cdots,n)$, $f'(x_0)$ 和 $f'(x_n)$ 的值即可.

计算压紧样条的 MATLAB 主程序

输入的量:$\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 分别是由节点 (x_i,y_i) $(i=0,1,2,\cdots,n)$ 的横坐标和纵坐标组成的向量, $dy0=f'(x_0)$, $dyn=f'(x_n)$.

输出的量:m 是 $\boldsymbol{X}$ 的维数, $\boldsymbol{H}$, $lambda$, mu 和 $\boldsymbol{D}$ 分别是由 $h_{i-1}(i=1,2,\cdots,n)$, λ_i,μ_i 和 $d_i(i=1,2,\cdots,n-1)$ 的坐标组成的向量, $\boldsymbol{A}$ 是(6.90)式的系数矩阵, $\boldsymbol{dY}$ 是线性方程组(6.90)的解向量, s_k 是压紧样条函数.

```
function [m,H,lambda,mu,D,A,dY,sk] = ClampedSp(X,Y,dy0,dyn)
m = length(X);A = zeros(m,m); n = m - 1;
H = zeros(1,n);lambda = zeros(1,n);
mu = zeros(1,n);lambda(1) = 0;A(1,1) = 2;A(1,2) = lambda(1);
D = zeros(1,n);H(1) = X(2) - X(1);mu(1) = 1 - lambda(1);D(1) = 2 * dy0;
for k = 1:n
  hk = X(k + 1) - X(k);H(k + 1) = hk;
end
H = H(2:n + 1);
for k = 1:n - 1
lambdak = H(k) / (H(k) + H(k + 1)); lambda(k + 1) = lambdak;
muk = 1 - lambda(k + 1);mu(k) = muk;
dk = 3 * ((mu(k) .* (Y(k + 1) - Y(k)) ./ H(k)) + (lambda(k + 1) .* (Y(k
+ 2) - Y(k + 1)) ./ H(k + 1)));
D(k + 1) = dk;
end
```

```
D(m) = 2 * dyn;mu(n) = 0; n;H;lambda;mu;D;
for i = 1:m - 1
A(i,i) = 2;A(m,m) = 2; A(i,i + 1) = lambda(i); A(i + 1,i) = mu(i);
end
dY = A\D';
syms x
m = length(X);S = zeros(m - 1,1);
for k = 2:m
sk = Y(k - 1) * ((H(k - 1) - 2 * X(k - 1) + 2 * x) * (x - X(k))^2)/(H(k -
1)^3) + Y(k) * ((H(k - 1) + 2 * X(k) - 2 * x) * (x - X(k - 1))^2)/(H(k - 1)^3) +
dY(k - 1) * ((x - X(k - 1)) * (x - X(k))^2)/(H(k - 1)^2) + dY(k) * ((x - X(k))
* (x - X(k - 1))^2)/(H(k - 1)^2)
end
```

例 6.7.2 用计算压紧样条的 MATLAB 主程序 ClampedSp.m 求例 6.7.1 的问题.

解 保存 ClampedSpren.m 为 M 文件,输入程序

```
>> X = [0:2:6,10],Y = [0,16,36,54,82],dy0 = 8,dyn = 7,
[m,H,lambda,mu,D,A,dY,sk] = ClampedSp(X,Y,dy0,dyn)
```

运行后输出压紧样条函数 s_k,X 的维数 m,由 $h_{i-1}(i=1,2,\cdots,n)$,λ_i,μ_i 和 $d_i(i=1,2,\cdots,n-1)$的坐标组成的向量 H,$lambda$,mu 和 D,(6.90)式的系数矩阵 A,线性方程组(6.90)的解向量 dY 如下

```
sk =
2 * (6 - 2 * x) * x^2 + 2 * x * (x - 2)^2 + 9/4 * (x - 2) * x^2
sk =
2 * ( - 2 + 2 * x) * (x - 4)^2 + 9/2 * (10 - 2 * x) * (x - 2)^2 + 9/4 * (x -
2) * (x - 4)^2 + 5/2 * (x - 4) * (x - 2)^2
sk =
9/2 * ( - 6 + 2 * x) * (x - 6)^2 + 27/4 * (14 - 2 * x) * (x - 4)^2 + 5/2 *
(x - 4) * (x - 6)^2 + 2 * (x - 6) * (x - 4)^2
sk =
27/32 * ( - 8 + 2 * x) * (x - 10)^2 + 41/32 * (24 - 2 * x) * (x - 6)^2 + 1/
2 * (x - 6) * (x - 10)^2 + 7/16 * (x - 10) * (x - 6)^2
m =
    5
H =
    2    2    2    4
lambda =
```

```
            0    0.5000    0.5000    0.3333

mu =

       0.5000    0.5000    0.6667         0

D =

     16.0000   27.0000   28.5000   25.0000   14.0000

A -

      2.0000         0         0         0         0
      0.5000    2.0000    0.5000         0         0
           0    0.5000    2.0000    0.5000         0
           0         0    0.6667    2.0000    0.3333
           0         0         0         0    2.0000

dY =

      8.0000
      9.0000
     10.0000
      8.0000
      7.0000
```

输入程序

```
>> x =3,
s2 =2 * ( -2 +2 * x) * (x -4)^2 +9/2 * (10 -2 * x) * (x -2)^2 +9/4 *
(x -2) * (x -4)^2 +5/2 * (x -4) * (x -2)^2
x =8,
s4 =27/32 * ( -8 +2 * x) * (x -10)^2 +41/32 * (24 -2 * x) * (x -6)^2
+1/2 * (x -6) * (x -10)^2 +7/16 * (x -10) * (x -6)^2
```

运行后输出$f(3)\approx s_2(3)$和$f(8)\approx s_4(8)$的值如下

```
x =           s2 =          x =        s4 =
     3             25.7500       8            68.5000
```

例 6.7.2 和例 6.7.1 计算的结果完全相同.

根据推论6.5编写了下面的计算自然样条函数及其h_{i-1}，λ_i，μ_i，d_i和线性方程组(6.88)的解的 MATLAB 程序，读者使用时，只需输入节点(x_i,y_i) $(i=0,1,2,\cdots,n)$即可.

计算自然样条的 MATLAB 主程序

输入的量：X 和 Y 分别是由节点(x_i,y_i) $(i=0,1,2,\cdots,n)$的横坐标和纵坐标组成的向量.

输出的量：m 是 X 的维数，H，$lambda$，mu 和 D 分别是由 $h_{i-1}(i=1,2,\cdots,n)$，λ_i，μ_i 和 d_i $(i=1,2,\cdots,n-1)$的坐标组成的向量，A 是(6.88)式的系数矩阵，dY 是线性方程组(6.88)的解向量，s_k 是自然样条函数.

```
function [m,H,lambda,mu,D,A,dY,sk] = NaturalSp(X,Y)
m = length(X);A = zeros(m,m); n = m - 1;
H = zeros(1,n);lambda = zeros(1,n);
mu = zeros(1,n);lambda(1) = 1;A(1,1) = 2;A(1,2) = lambda(1);
D = zeros(1,n);H(1) = X(2) - X(1);mu(1) = 1;D(1) = 3 * (Y(2) - Y(1));
for k = 1:n
  hk = X(k + 1) - X(k);H(k + 1) = hk;
end
H - H(2:n + 1);
for k = 1:n - 1
lambdak = H(k)/(H(k) + H(k + 1)); lambda(k + 1) = lambdak;
muk = 1 - lambda(k + 1);mu(k) = muk;
dk = 3 * ((mu(k).* (Y(k + 1) - Y(k))./H(k)) + (lambda(k + 1).* (Y(k
+2) - Y(k + 1))./H(k + 1)));
D(k + 1) = dk;
end
D(m) = 3 * (Y(m) - Y(m - 1))/H(m - 1);mu(n) = 1; n;H;lambda;mu;D;
for i = 1:m - 1
A(i,i) = 2;A(m,m) = 2; A(i,i + 1) = lambda(i); A(i + 1,i) = mu(i);
end
dY = A\D';
syms x
m = length(X);S = zeros(m - 1,1);
for k = 2:m
sk = Y(k - 1) * ((H(k - 1) - 2 * X(k - 1) + 2 * x) * (x - X(k))^2)/(H(k -
1)^3) + Y(k) * ((H(k - 1) + 2 * X(k) - 2 * x) * (x - X(k - 1))^2)/(H(k - 1)^3) +
dY(k - 1) * ((x - X(k - 1)) * (x - X(k))^2)/(H(k - 1)^2) + dY(k) * ((x - X(k))
* (x - X(k - 1))^2)/(H(k - 1)^2)
end
```

例 6.7.3 用计算自然样条的 MATLAB 主程序 NaturalSp.m 求例 6.7.1 的问题.

解 保存 NaturalSp.m 为 M 文件,输入程序

```
>>  X = [0:2:6,10],Y = [0,16,36,54,82],dy0 = 8,dyn = 7,
[m,H,lambda,mu,D,A,dY,sk] = NaturalSp (X,Y)
```

运行后输出自然样条函数 s_k, $\boldsymbol{X}$ 的维数 m,由 h_{i-1} $(i=1,2,\cdots,n)$,λ_i,μ_i 和 d_i $(i=1,2,\cdots,n-1)$的坐标组成的向量 $\boldsymbol{H}$,*lambda*,*mu* 和 $\boldsymbol{D}$,(6.88)式的系数矩阵 $\boldsymbol{A}$,线性方程组(6.88)的解向量 $\boldsymbol{dY}$ 如下

```
sk =
```

```
2*(6 -2*x)*x^2 +915/172*x*(x -2)^2 +117/86*(x-2)*x^2
sk =
2*( -2 +2*x)*(x-4)^2 +9/2*(10 -2*x)*(x-2)^2 +117/86*
(x-2)*(x-4)^2 +471/172*(x-4)*(x-2)^2
sk =
9/2*( -6 +2*x)*(x-6)^2 +27/4*(14 -2*x)*(x-4)^2 +471/
172*(x-4)*(x-6)^2 +333/172*(x-6)*(x-4)^2
sk =
27/32*( -8 +2*x)*(x-10)^2 +41/32*(24 -2*x)*(x-6)^2 +
333/688*(x-6)*(x-10)^2 +285/688*(x-10)*(x-6)^2
m =      5
H =      2          2          2          4
lambda =      1          1/2          1/2          1/3
mu =    1/2          1/2          2/3          1
D =    48          27          57/2          25          21
A =

        2          1          0          0          0
      1/2          2        1/2          0          0
        0        1/2          2        1/2          0
        0          0        2/3          2        1/3
        0          0          0          1          2

dY =
  915/43
  234/43
  471/43
  333/43
  285/43
```

输入程序

```
>> x=3,
s2 =2*( -2 +2*x)*(x-4)^2 +9/2*(10 -2*x)*(x-2)^2 +117/
86*(x-2)*(x-4)^2 +471/172*(x-4)*(x-2)^2
x=8,
s4 =s4 =27/32*( -8 +2*x)*(x-10)^2 +41/32*(24 -2*x)*(x-
6)^2 +333/688*(x-6)*(x-10)^2 +285/688*(x-10)*(x-6)^2
```

运行后输出 $f(3)\approx s_2(3)$ 和 $f(8)\approx s_4(8)$ 的值如下

```
x =     s2 =               x =            s4 =
 3       24.6221             8               68.5581
```

例 6.7.3 和例 6.7.2 用的方法不同,所以计算的结果不同,但是两种方法计

算的结果相近.

6.7.5 用二阶导数计算的几种样条函数

上节给出了用定理6.11的(6.65)、(6.66)计算样条函数的公式.下面由推论6.3的(6.72)、(6.73)和在给定了表6-7中六种端点约束条件之一时,可以证明 $S_n(x)$ 被唯一确定,同时也推导出用二阶导数计算这几种样条函数的计算公式.

推论6.7(外推样条) 如果取被插值函数 $y=f(x)$ 的端点约束条件为 $S''_n(x_0)=S''_n(x_1)-\frac{h_0}{h_1}[S''_n(x_2)-S''_n(x_1)]$, $S''_n(x_n)=S''_n(x_{n-1})-\frac{h_{n-1}}{h_{n-2}}[S''_n(x_{n-1})-S''_n(x_{n-2})]$,则存在唯一的 $f(x)$ 的三次样条函数 $S_n(x)$,且 $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上的三次多项式 $s_i(x)$ $(i=1,2,\cdots,n)$ 为(6.72)式,即

$$S_n(x)=s_i(x)=\frac{S''_n(x_{i-1})}{6h_{i-1}}(x_i-x)^3-\frac{S''_n(x_i)}{6h_{i-1}}(x_{i-1}-x)^3+$$
$$p_{i-1}(x_i-x)-q_{i-1}(x_{i-1}-x),$$

其中 $S''_n(x_i)$ 和 $S''_n(x_{i-1})$ $(i=1,2,\cdots,n)$ 是线性方程组

$$\begin{cases}\left(3h_0+2h_1+\dfrac{h_0^2}{h_1}\right)S''_n(x_1)+\left(h_1-\dfrac{h_0^2}{h_1}\right)S''_n(x_2)=u_1,\\ h_{i-1}S''_n(x_{i-1})+2(h_{i-1}+h_i)S''_n(x_i)+h_iS''_n(x_{i+1})=u_i\quad(i=2,3,\cdots,n-2),\\ \left(h_{n-2}-\dfrac{h_{n-1}^2}{h_{n-2}}\right)S''_n(x_{n-2})+\left(2h_{n-2}+3h_{n-1}+\dfrac{h_{n-1}^2}{h_{n-2}}\right)S''_n(x_{n-1})=u_{n-1}\end{cases}\tag{6.91}$$

的解,且(6.91)有且只有唯一解.其中 $h_{i-1}=x_i-x_{i-1}$, $p_i=\frac{y_i}{h_i}-\frac{S''_n(x_i)}{6}h_i$, $q_i=\frac{y_{i+1}}{h_i}-\frac{S''_n(x_{i+1})}{6}h_i$, $b_i=\frac{y_{i+1}-y_i}{h_i}$, $u_i=6(b_i-b_{i-1})$ $(i=1,2,\cdots,n)$.

证明 根据推论6.3可知外推样条为(6.72)式,且 $S''_n(x_i)$ 和 $S''_n(x_{i-1})$ $(i=1,2,\cdots,n)$ 是(6.91)的解.由于(6.91)式的系数矩阵是严格对角占优,所以(6.91)式的系数矩阵是非奇异矩阵.因此矩阵方程(6.91)有且只有唯一解.

注意 外推样条等价于端点 x_0,x_n 处的三次多项式曲线是由相邻区间上的三次多项式曲线的扩展形成的样条(见例6.7.4).

定理6.13 如果 $S_n(x)$ 是定义6.5定义的 $y=f(x)$ 在点 (x_i,y_i) $(i=0,1,2,\cdots,n)$ 处的三次样条函数,给定了表6-7中六种端点约束条件之一,则存在唯一的 $f(x)$ 的三次样条函数 $S_n(x)$,且 $S_n(x)$ 在每个子区间 $[x_{i-1},x_i]$ 上的三次多项式 $s_i(x)$ $(i=1,2,\cdots,n)$ 为

$$s_i(x)=c_{i,3}(x-x_{i-1})^3+c_{i,2}(x-x_{i-1})^2+c_{i,1}(x-x_{i-1})+c_{i,0},$$

其中系数为

$$c_{i,0}=y_{i-1},\quad c_{i,1}=\frac{y_i-y_{i-1}}{h_{i-1}}-\frac{h_{i-1}}{6}[2S_n''(x_{i-1})+S_n''(x_i)],$$

$$c_{i,2}=\frac{S_n''(x_{i-1})}{2},\qquad c_{i,3}=\frac{1}{6h_{i-1}}[S_n''(x_i)-S_n''(x_{i-1})]. \tag{6.92}$$

而 $S_n''(x_i)$ 和 $S_n''(x_{i-1})$ $(i=1,2,\cdots,n)$ 可以由线性方程组(6.73)，即

$$h_{i-1}S_n''(x_{i-1})+2(h_{i-1}+h_i)S_n''(x_i)+h_iS_n''(x_{i+1})$$

$$=6\left(\frac{y_{i+1}-y_i}{h_i}-\frac{y_i-y_{i-1}}{h_{i-1}}\right)(i=1,2,\cdots,n) \tag{6.93}$$

与下面给出表 6-8 所示的六种样条端点约束条件之一的方程组联立，解出唯一解 $S_n''(x_i)$ 和 $S_n''(x_{i-1})$.

表 6-8 端点约束条件包含 $S_n''(x_0)$ 和 $S_n''(x_n)$ 的方程

序号	样条种类	包含 $S_n''(x_0)$ 和 $S_n''(x_n)$ 的方程
1	压紧样条 (Clamped Spline) (已知一阶导数)	$S_n''(x_0)=\frac{3}{h_0}\left[\frac{y_1-y_0}{h_0}-f'(x_0)\right]-\frac{S_n''(x_1)}{2}$ $S_n''(x_0)=-\frac{3}{h_{n-1}}\left[\frac{y_n-y_{n-1}}{h_{n-1}}-f'(x_n)\right]-\frac{S_n''(x_{n-1})}{2}$
2	周期端点样条 $f(x_0)=f(x_n)$	$S_n(x_0)=S_n(x_n),S_n'(x_0)=S_n'(x_n),$ $S_n''(x_0)=S_n''(x_n)$
3	自然样条 (Natural Spline)	$S_n''(x_0)=0,$ $S_n''(x_n)=0$
4	外推样条 (Extrapolated Spline)	$S_n''(x_0)=S_n''(x_1)-\frac{h_0}{h_1}[S_n''(x_2)-S_n''(x_1)]$ $S_n''(x_n)=S_n''(x_{n-1})+\frac{h_{n-1}}{h_{n-2}}[S_n''(x_{n-1})-S_n''(x_{n-2})]$
5	抛物线结尾样条 (Parabolically Terminated Spline)	$(3h_0+2h_1)S_n''(x_1)+h_1S_n''(x_2)=u_1$ $(2h_{n-2}+3h_{n-1})S_n''(x_{n-1})+h_{n-2}S_n''(x_{n-2})=u_{n-1}$
6	端点调整曲率样条 (End-point Curvature-adjusted Spline)	$h_0S_n''(x_0)=u_1-2(h_0+h_1)S_n''(x_1)-h_1S_n''(x_2)$ $h_{n-1}S_n''(x_n)=u_{n-1}-2(h_{n-2}+h_{n-1})S_n''(x_{n-1})-h_{n-2}S_n''(x_{n-2})$

例 6.7.4 求经过点$(0,0),\left(1,\frac{1}{2}\right),(2,2),\left(3,\frac{3}{2}\right)$的外推样条函数.

解 (1) 根据推论 6.7 计算

$$h_{i-1}=x_i-x_{i-1}=1,\ i=1,2,3,$$

$$b_0=\frac{y_1-y_0}{h_0}=\frac{1}{2},\ b_1=\frac{y_2-y_1}{h_1}=\frac{3}{2},\ b_2=\frac{y_3-y_2}{h_2}=-\frac{1}{2},$$

$$u_1=6(b_1-b_0)=6,\ u_2=6(b_2-b_1)=-12,$$

(2) 根据(6.91)式,得线性方程组

$$\begin{cases}\left(3h_0+2h_1+\dfrac{h_0^2}{h_1}\right)S_3''(x_1)+\left(h_1-\dfrac{h_0^2}{h_1}\right)S_3''(x_2)=u_1,\\ \left(h_1-\dfrac{h_2^2}{h_1}\right)S_3''(x_1)+\left(2h_1+3h_2+\dfrac{h_2^2}{h_1}\right)S_3''(x_2)=u_2,\end{cases}$$

其矩阵方程为

$$\begin{pmatrix}6&0\\0&6\end{pmatrix}\begin{pmatrix}S_3''(x_1)\\S_3''(x_2)\end{pmatrix}=\begin{pmatrix}6\\-12\end{pmatrix}.$$

解得 $S_3''(x_1)=1,S_3''(x_2)=-2.$

(3) 将 $S_3''(x_1)=1,S_3''(x_2)=-2$ 代入端点约束条件

$$S_n''(x_0)=S_n''(x_1)-\frac{h_0}{h_1}[S_n''(x_2)-S_n''(x_1)],$$

$$S_n''(x_n)=S_n''(x_{n-1})-\frac{h_{n-1}}{h_{n-2}}[S_n''(x_{n-1})-S_n''(x_{n-2})]$$

中,得 $S_3''(x_0)=4,S_3''(x_3)=-5.$

$$p_0=\frac{y_0}{h_0}-\frac{S_n''(x_0)}{6}h_0=-\frac{2}{3},\ q_0=\frac{y_1}{h_0}-\frac{S_n''(x_1)}{6}h_0=\frac{1}{3},$$

$$p_1=\frac{y_1}{h_1}-\frac{S_n''(x_1)}{6}h_1=\frac{1}{3},q_1=\frac{y_2}{h_1}-\frac{S_n''(x_2)}{6}h_1=\frac{7}{3},$$

$$p_2=\frac{y_2}{h_2}-\frac{S_n''(x_2)}{6}h_2=\frac{7}{3},\ q_2=\frac{y_3}{h_2}-\frac{S_n''(x_3)}{6}h_2=\frac{7}{3},$$

(4) 将(3)的计算结果代入(6.72)式计算样条的系数可得三次样条 $S_n(x)=\{s_i(x),\ x\in[x_{i-1},x_i],i=1,2,3\}$ 为

$$s_1(x)=\frac{S_n''(x_0)}{6h_0}(x_1-x)^3-\frac{S_n''(x_1)}{6h_0}(x_0-x)^3+p_0(x_1-x)-q_0(x_0-x),\quad 0\leqslant x\leqslant 1,$$

$$s_2(x)=\frac{S_n''(x_1)}{6h_1}(x_2-x)^3-\frac{S_n''(x_2)}{6h_1}(x_1-x)^3+p_1(x_2-x)-q_1(x_1-x),\quad 1\leqslant x\leqslant 2,$$

$$s_3(x)=\frac{S_n''(x_2)}{6h_2}(x_3-x)^3-\frac{S_n''(x_3)}{6h_2}(x_2-x)^3+p_2(x_3-x)-q_2(x_2-x),\quad 2\leqslant x\leqslant 3,$$

即

$$S_n(x)=\begin{cases}\dfrac{2}{3}(1-x)^3+\dfrac{x^3}{6}-\dfrac{2}{3}(1-x)+\dfrac{1}{3}, & 0\leqslant x\leqslant 1,\\ \dfrac{1}{6}(2-x)^3+\dfrac{1}{3}(1-x)^3+\dfrac{1}{3}(2-x)-\dfrac{7}{3}(1-x), & 1\leqslant x\leqslant 2,\\ -\dfrac{1}{3}(3-x)^3+\dfrac{5}{6}(2-x)^3+\dfrac{7}{3}(3-x)-\dfrac{7}{3}(2-x), & 2\leqslant x\leqslant 3.\end{cases}$$

例 6.7.5 求经过点(0,0),(1,0.5),(2,2.0),(3,1.5)的抛物线结尾样条函数.

解 (1) 将例6.7.4中计算的值 $h_{i-1}=x_i-x_{i-1}=1,i=1,2,3,b_0=0.5,b_1=1.5,b_2=-0.5,u_1=6.0,u_2=-12.0$ 代入表6-8中抛物线结尾样条的端点约束条件($n=3$)

$$(3h_0+2h_1)S_n''(x_1)+h_1S_n''(x_2)=u_1$$
$$(2h_{n-2}+3h_{n-1})S_n''(x_{n-1})+h_{n-2}S_n''(x_{n-2})=u_{n-1},$$

即

$$(3+2)S_n''(x_1)+S_n''(x_2)=6.0,$$
$$(2+3)S_3''(x_2)+S_3''(x_1)=-12.0.$$

其矩阵方程为

$$\begin{pmatrix}5.0 & 1.0\\ 1.0 & 5.0\end{pmatrix}\begin{pmatrix}S_3''(x_1)\\ S_3''(x_2)\end{pmatrix}=\begin{pmatrix}6.0\\ -12.0\end{pmatrix}.$$

解得 $S_3''(x_1)=1.75,S_3''(x_2)=-2.75$.

(2) 由于在每个端点的子区间[0,1],[2,3]上 $S_n'''(x)\equiv 0$,从而 $S_n(x)$ 的二阶导函数 $S_n''(x)$ 在[0,1],[2,3]上分别为常数,所以取

$$S_3''(x_0)=S_3''(x_1)=1.75,\quad S_3''(x_3)=S_3''(x_2)=-2.75.$$

(3) 将 $S_3''(x_i)(i=0,1,2,3)$ 的值代入(6.92)式计算样条的系数可得三次样条为

$$s_1(x)=0.875x^2-0.375x,\qquad 0\leqslant x\leqslant 1,$$
$$s_2(x)=0.75(1-x)^3+0.875(1-x)^2-1.375(1-x)+0.5,\quad 1\leqslant x\leqslant 2,$$
$$s_3(x)=-1.375(2-x)^2-0.875(2-x)+2.0,\qquad 2\leqslant x\leqslant 3,$$

即

$$S_n(x)=\begin{cases}0.875x^2-0.375x, & 0\leqslant x\leqslant 1,\\ 0.75(1-x)^3+0.875(1-x)^2-1.375(1-x)+0.5, & 1\leqslant x\leqslant 2,\\ -1.375(2-x)^2-0.875(2-x)+2.0, & 2\leqslant x\leqslant 3.\end{cases}$$

由计算结果可见,抛物线结尾样条函数在每个端点的子区间[0,1],[2,3]上退化为二次函数.

6.7.6 用 MATLAB 计算三次样条

分段三次样条插值的计算在 MATLAB 函数库中有现成的程序，在计算时只要直接调用即可. 计算分段三次样条插值的 MATLAB 程序分为两类：一类是默认端点约束条件的 MATLAB 程序，即不需要用户输入端点约束条件，程序运算时自动选取，进行运算；另一类是给定端点约束条件的 MATLAB 程序，即需要用户输入端点约束条件，程序根据输入的端点约束条件进行运算. 详细的调用方法见表 6－9、表 6－10 和例题.

根据上节定理和推论编写的计算压紧样条和所有的其他种类三次样条插值的 MATLAB 程序为 spline.m，输入端点约束条件的 MATLAB 程序的详细地调用方法参考下面的表 6－9 和例题.

表 6－9 计算各种三次样条的 MATLAB 命令

输入端点约束条件的 MATLAB 命令	功 能
YI = spline (X, [df0,Y,dfn],XI)	spline (X, [df0,Y,dfn],XI) 命令的主要功能是通过用户输入端点约束条件，可以进行计算函数 $\boldsymbol{Y}$ 在插值点向量 $\boldsymbol{X}_i$ 的元素处的压紧样条和所有的三次样条内插值所对应的向量 $\boldsymbol{Y}_i$. 如果输入的节点 $(\boldsymbol{X}(i),\boldsymbol{Y}(i))$ 的横坐标向量 $\boldsymbol{X}$ 和纵坐标向量 $\boldsymbol{Y}$ 都是 n 维向量，df_0 和 df_n 分别是三次样条端点约束条件中插值区间端点的导数 $S_n'(x_0)=f'(x_0)$，$S_n'(x_n)=f'(x_n)$，插值点向量 $\boldsymbol{X}_i$ 是 m 维，则运行后输出的插值向量 $\boldsymbol{Y}_i$ 也是 m 维.
YI = spline (X, Y ,XI)	如果输入的节点 $(\boldsymbol{X}(i),\boldsymbol{Y}(i))$ 的横坐标向量 $\boldsymbol{X}$ 是 n 维，而纵坐标向量 $\boldsymbol{Y}$ 是 $n+2$ 维，即 $\boldsymbol{Y}$ 比 $\boldsymbol{X}$ 多两个元素，则 $\boldsymbol{Y}$ 的第一个和最后一个元素被用作三次样条端点约束条件中插值区间端点的导数，即 f(X) = Y(:,2 :end 1), df(min(X)) = Y(:,1), df(max(X)) = Y(:,end) 通过用户输入端点约束条件进行计算函数 $\boldsymbol{Y}$ 在插值点向量 $\boldsymbol{X}_i$ 的元素处的压紧样条或各种三次样条内插值所对应的向量 $\boldsymbol{Y}_i$.
PP = spline (X, Y) 或 PP = spline (X,[df0,Y,dfn])	如果只输入节点 $(\boldsymbol{X}(i),\boldsymbol{Y}(i))$ 的横坐标向量 $\boldsymbol{X}$ 和纵坐标向量 $\boldsymbol{Y}$，或将 $\boldsymbol{Y}$ 用 [df0,Y,dfn] 替换，则运行后输出三次样条插值的分段多项式的构成信息.

例 6.7.6 给出节点的数据如下：

x	−1.00	−0.54	0.13	1.12	1.89	2.06	2.54	2.82	3.50
$f(x)$	−2.46	−5.26	−1.87	0.05	1.65	2.69	4.56	7.89	10.31

分别求在下列条件下在插值点 $x_1=-0.02, x_2=2.56$ 处的压紧三次样条插值，并显示该样条函数的有关信息：

(1) 端点约束条件为 $S_n'(-1)=5, S_n'(3.50)=29.16$；

(2) 端点约束条件为 $S_n'(-1)=0, S_n'(3.50)=0$.

解 (1) 输入 MATLAB 程序

```
>> X=[-1.00  -0.54  0.13  1.12  1.89  2.06  2.54  2.82  3.50];
   Y=[-2.46  -5.26  -1.87  0.05  1.65  2.69  4.56  7.89
10.31]; XI=[-0.02  2.56];
   YI=spline(X,[5,Y,29.16],XI), PP=spline(X,[5,Y,29.16])
```

运行后屏幕显示压紧样条分别在 $x_1=-0.02, x_2=2.56$ 处的插值和该样条函数的有关信息如下

```
YI =
    -3.1058    4.7834
PP =
    form: 'pp'
    breaks: [-1 -0.5400 0.1300 1.1200 1.8900 2.0600 2.5400 2.8200
3.5000]
    coefs: [8x4 double]
    pieces: 8
    order: 4
    dim: 1
```

(2) 因为端点约束条件为 $S_n'(-1)=0, S_n'(3.50)=0$，所以输入 MATLAB 程序

```
>> YI=spline(X,[0,Y,0],XI), PP=spline(X,[0,Y,0])
```

运行后屏幕显示压紧三次样条分别在 $x_1=-0.02, x_2=2.56$ 的插值和该样条函数的有关信息如下

```
YI =
    -3.0192    4.7501
PP =
      form: 'pp'
    breaks: [-1 -0.5400 0.1300 1.1200 1.8900 2.0600 2.5400 2.8200
3.5000]
    coefs: [8x4 double]
```

```
pieces: 8
order: 4
  dim: 1
```

另一类调用 MATLAB 程序计算分段三次样条数据插值的方法是用户不需要输入端点约束条件，程序运算时自动选取，进行运算. 这类默认端点约束条件的 MATLAB 程序的详细地调用方法参考下面的表 6－10 和例题.

表 6－10 计算各种三次样条的 MATLAB 命令

默认端点约束条件的 MATLAB 命令	功能
YI = interp1(X,Y,XI,'spline') 或用 YI = spline(X,Y,XI)	interp1(X,Y,XI,'spline')或 spline(X,Y,XI)命令的主要功能是计算函数 Y 在插值点向量 X_i 的元素处的三次样条内插值所对应的向量 Y_i (1) 如果输入的节点($X(i)$,$Y(i)$)的横坐标向量 X 和纵坐标向量 Y 都是 n 维向量，插值点向量 X_i 是 m 维，则运行后输出的插值向量 Y_i 也是 m 维 (2) 如果输入的节点($X(i)$,$Y(i)$)的横坐标向量 X 是 n 维，纵坐标矩阵 $Y_{n\times t}$ 的行数与 X 的维数 n 相同，插值点向量 X_i 是 m 维，则按矩阵 Y 的每列进行插值运算，运行后输出的插值矩阵 Y_i 是 m 行 t 列
YI = interp1(Y,XI,'spline')	interp1(Y,XI,'spline')或 spline(Y,XI)命令与查表的作用相同. 这表指的是[X,Y]，要查在 X 的元素之间位置的 X_i 的元素的三次样条内插值 Y_i 的元素，即返回值 interp1(Y,XI)命令主要用于节点($X(i)$,$Y(i)$)的横坐标向量 X 的元素是 1 到 n 的自然数. 由于interp1(Y,XI,'pchip')是 interp1(X,Y,XI,'pchip')的特例，所以两者输入和输出的向量或矩阵的大小类似

例 6.7.7 在默认端点约束条件下，用两种方法计算在下列插值点处的三次样条插值.

(1) 给出节点的数据与例 6.7.6 相同，插值点 $X_i = 2.56$;

(2) 节点($X(i)$,$Y(i)$)的横坐标向量 X 从 －5 到 5 的整数，纵坐标向量 Y = (－2.36,0.85,1.12,2.36,2.36,3,4,1.46,0.49,0.06,0)，插值点向量 XI 是从 －4 到 4 的 11 个等分点.

解 (1) 输入 MATLAB 程序

```
>>X=[-1.00  -0.54  0.13  1.12  1.89  2.06  2.54  2.82  3.50];
Y=[-2.46  -5.26  -1.87  0.05  1.65  2.69  4.56  7.89  10.31];
XI=2.56;
Y1=spline (X, Y,XI),Y2=interp1(X,Y,XI,'spline')
```

运行后屏幕显示三次样条插值的两种结果如下

```
Y1=4.730 2,Y2=4.730 2.
```

(2) 输入 MATLAB 程序

```
>> X= -5:5; Y=[-2.36 .85 1.12 2.36 2.36 3 4 1.46 .49 .06 0];
XI=linspace(-4,4,11); Y1=spline (X, Y,XI),
Y2=interp1(X,Y,XI,'spline')
```

运行后屏幕显示

```
Y1 =
    0.8500    1.0203    1.8857    2.4779    2.3683    3.0000
    4.0656    2.5935    0.8247    0.4074    0.0600
Y2 =
    0.8500    1.0203    1.8857    2.4779    2.3683    3.0000
    4.0656    2.5935    0.8247    0.4074    0.0600
```

例 6.7.8 设函数 $f(x)=\dfrac{1}{1+25x^2}$ 定义在区间 $[-1,1]$ 上，取 $n=10$，按等距节点构造分段三次样条函数 $S_n(x)$. 试用 MATLAB 程序计算在各小区间中点处分段三次样条插值 $S_n(x_{i+1/2})$ 及其相对误差.

解 记节点的横坐标 $x_i=-1+ih$, $h=0.2$ $(i=0,1,2,\cdots,10)$，插值点 $x_{i+\frac{1}{2}}=\dfrac{1}{2}(x_i+x_{i+1})$ $(i=0,1,2,\cdots,9)$. $S_{10}(x)$ 在小区间 $[x_i,x_{i+1}]$ $(i=0,1,2,\cdots,9)$ 中点 $x_{i+\frac{1}{2}}=\dfrac{1}{2}(x_i+x_{i+1})$ $(i=0,1,2,\cdots,9)$ 处的三次样条插值

$$S_{10}(x_{i+\frac{1}{2}})=\frac{1}{2}[f(x_i)+f(x_{i+1})],\ i=0,1,2,\cdots,9$$

的相对误差公式可以表示为

$$|R_{10}(x_{i+\frac{1}{2}})|=\left|\frac{f(x_{i+\frac{1}{2}})-S_{10}(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right|.$$

下面用 MATLAB 程序计算各小区间中点处插值 $S_{10}(x_{i+\frac{1}{2}})$ 及其相对误差 $|R_{10}(x_{i+\frac{1}{2}})|$ 的值. 在 MATLAB 工作窗口输入程序

```
>> h=0.2;x0= -1:h:1;y0=1./(1+25.*x0.^2);xi= -0.9:h:0.9;
fi=1./(1+25.*xi.^2); yi=spline (x0,y0,xi);
Ri=abs((fi-yi)./fi); xi,fi,yi,Ri,i=[xi',fi',yi',Ri']
```

运行后屏幕显示各小区间中点 $x_i = x_{i+\frac{1}{2}}$ 处的函数值 $f_i = f(x_{i+\frac{1}{2}})$，插值 $S_i = S_{10}(x_{i+\frac{1}{2}})$，相对误差值 $R_i = \left|\frac{f(x_{i+\frac{1}{2}}) - S_{10}(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right|$，经过整理后列入表 6－11 中.

表 6－11 例 6.7.8 的各小区间中点 x_i 处的函数值 f_i，三次样条插值 s_i 和相对误差值 R_i

i	$x_{i+\frac{1}{2}} = \frac{1}{2}(x_i + x_{i+1})$	$f(x_{i+\frac{1}{2}})$	$S_{10}(x_{i+\frac{1}{2}})$	$\left\|\frac{f(x_{i+\frac{1}{2}}) - S_{10}(x_{i+\frac{1}{2}})}{f(x_{i+\frac{1}{2}})}\right\|$
0	−0.900 000 000 000 00	0.047 058 823 529 41	0.048 370 807 482 39	0.027 879 659 000 79
1	−0.700 000 000 000 00	0.075 471 698 113 21	0.074 479 871 250 64	0.013 141 705 929 00
2	−0.500 000 000 000 00	0.137 931 034 482 76	0.140 135 046 881 56	0.015 979 089 891 31
3	−0.300 000 000 000 00	0.307 692 307 692 31	0.297 332 882 399 59	0.033 668 132 201 33
4	−0.100 000 000 000 00	0.800 000 000 000 00	0.820 533 423 520 08	0.025 666 779 400 10
5	0.100 000 000 000 00	0.800 000 000 000 00	0.820 533 423 520 08	0.025 666 779 400 10
6	0.300 000 000 000 00	0.307 692 307 692 31	0.297 332 882 399 59	0.033 668 132 201 33
7	0.500 000 000 000 00	0.137 931 034 482 76	0.140 135 046 881 56	0.015 979 089 891 31
8	0.700 000 000 000 00	0.075 471 698 113 21	0.074 479 871 250 64	0.013 141 705 929 00
9	0.900 000 000 000 00	0.047 058 823 529 41	0.048 370 807 482 39	0.027 879 659 000 79

6.7.7 几种作三次样条有关图像的 MATLAB 程序

如果能在同一个坐标系中作出在插值区间 $[a,b]$ 上的节点 $(x_i, f(x_i))$，被插值函数 $f(x)$，$f(x)$ 的分段三次样条函数 $S_n(x)$ 和插值点 $(x_j, S_n(x_j))$ 等图像，则更有利于我们进一步直观地研究我们关心的问题. 为此编写了下面几种 MATLAB 程序，读者可以根据各自的需要选择不同的程序.

求有关分段三次样条图形的 MATLAB 主程序

输入的量：$n+1$ 个节点 $(x_i, f(x_i))$ $(i = 1, 2, \cdots, n+1)$ 横坐标组成的数组 x_0，纵坐标组成的数组 y_0，插值点 $(x_j, S_n(x_j))$ 横坐标组成的数组 x_j 和在 x 处函数 $y = f(x)$ 的值，dy_1 和 dy_n 是端点约束条件.

输出的量：插值 $S = S_n(x_j)$ 和 $S_n = S_n(x)$.

输出的图形：在插值区间 $[a,b]$ 上的节点 $(x_i, f(x_i))$，被插值函数 $f(x)$，$f(x)$ 的分段三次样条函数 $S_n(x)$ 和插值点 $(x_j, S_n(x_j))$.

使用说明：(1) 如果用户限定端点约束条件，则请用 splinetx.m 程序；

(2) 如果用户不限定端点约束条件，则请用 splinetx1.m 程序；

(3) 如果用户想作出与图 6－17 至 19 类似的图形，请参考自由作图程序.

(一) 限定端点约束条件的作图程序

```
function S = splinetx(x0,y0,xj,x,y,dy1,dyn)
S = spline(x0,[dy1,y0,dyn],xj);
Sn = spline(x0,[dy1,y0,dyn],x);
plot(x0,y0,'o',x,Sn,'-',xj,S,'*',x,y,'-.')
legend('节点(xi,yi)','分段三次样条函数','插值点(x,S)','被插值函数y')
```

(二) 不限定端点约束条件的作图程序

```
function S = splinetx1(x0,y0,xi,x,y)
S = interp1(x0,y0,xi,'spline');
Sn = interp1(x0,y0,x,'spline');
plot(x0,y0,'o',x,Sn,'-',xi,S,'*',x,y,'-.')
legend('节点(xi,yi)','分段三次样条函数','插值点(x,S)','被插值函数y')
```

(三) 自由作图程序

我们也可以直接在 MATLAB 工作窗口编程序,例如,

```
>> x1 = -8:4/3:-4,c1 = cos(x1);xx1 = -8:0.1:-4;
Y1 = interp1(x1,c1,xx1,'pchip');
pp1 = interp1 (x1,c1,xx1,'spline');
cc1 = cos(xx1);% pp1 = spline (x1,c1,xx1);
plot(x1,c1,'bo',xx1,pp1,'k-',xx1,Y1,'r:',xx1,cc1,'g-.')
subplot(2,2,2)
x2 = -4:4/3:-0;c2 = cos(x2); xx2 = -4:0.1:0;
pp2 = interp1 (x2,c2,xx2,'spline');
Y2 = interp1(x2,c2,xx2,'pchip');cc2 = cos(xx2);
plot(x2,c2,'bo',xx2,pp2,'k-',xx2,Y2,'r:',xx2,cc2,'g-.')
subplot(2,1,2)
x = -8:4/3:8;c = cos(x);xx = -8:0.1:8;
pp = spline(x,c,xx);Y = interp1(x,c,xx,'pchip');
cc = cos(xx);
plot(x,c,'bo',xx,pp,'k-',xx,Y,'r:',xx,cc,'g-.')
legend('节点(xi,yi)','三次样条插值函数','分段三次埃尔米特插值','y =
cosx 的函数')
```

运行后屏幕显示图 6-18.

例 6.7.9 设函数 $f(x)=\dfrac{1}{1+x^2}$ 定义在区间 $[-5,5]$ 上,节点 $(X(i), f(X(i)))$ 的横坐标向量 X 的元素是首项 $a=-5$,末项 $b=5$,公差 $h=1.5$ 的等差数列,构造分段三次样条函数 $S_n(x)$. 把区间 $[-5,5]$ 分成 20 等份,构成 20 个小区间,用限定端点约束条件和不限定端点约束条件的 MATLAB 程序计算各小

区间中点 $x_{i+\frac{1}{2}}$ 处 $S_n(x)$ 的值，并作出节点，插值点，$f(x)$ 和 $S_n(x)$ 的图形，并与分段埃尔米特插值函数的图 6－14 比较.

解 记 $x_i=-5+ih, h=0.5, i=0,1,2,\cdots,20$，插值点 $x_{i+\frac{1}{2}}=\frac{1}{2}(x_i+x_{i+1}), i=0,1,2,\cdots,19$.

（1）不限定端点约束条件，在 MATLAB 工作窗口输入程序

```
>>x0 = -5:1.5:5;
y0 =1./(1 +x0.^2);
x1 = -4.75:0.5:4.75;
x = -5:0.001:5;
y =1./(1 +x.^2);
S =splinetx1(x0,y0,x1,x,y)
title('y =1/(1 +x^2)及其三次样条插值函数,节点和插值点的图形')
```

运行后屏幕显示各小区间中点 $x_{i+\frac{1}{2}}$ 处 $S_n(x)$ 的值，作出节点，插值点，$f(x)$ 和 $S_n(x)$ 的图形（见图 6－20）.

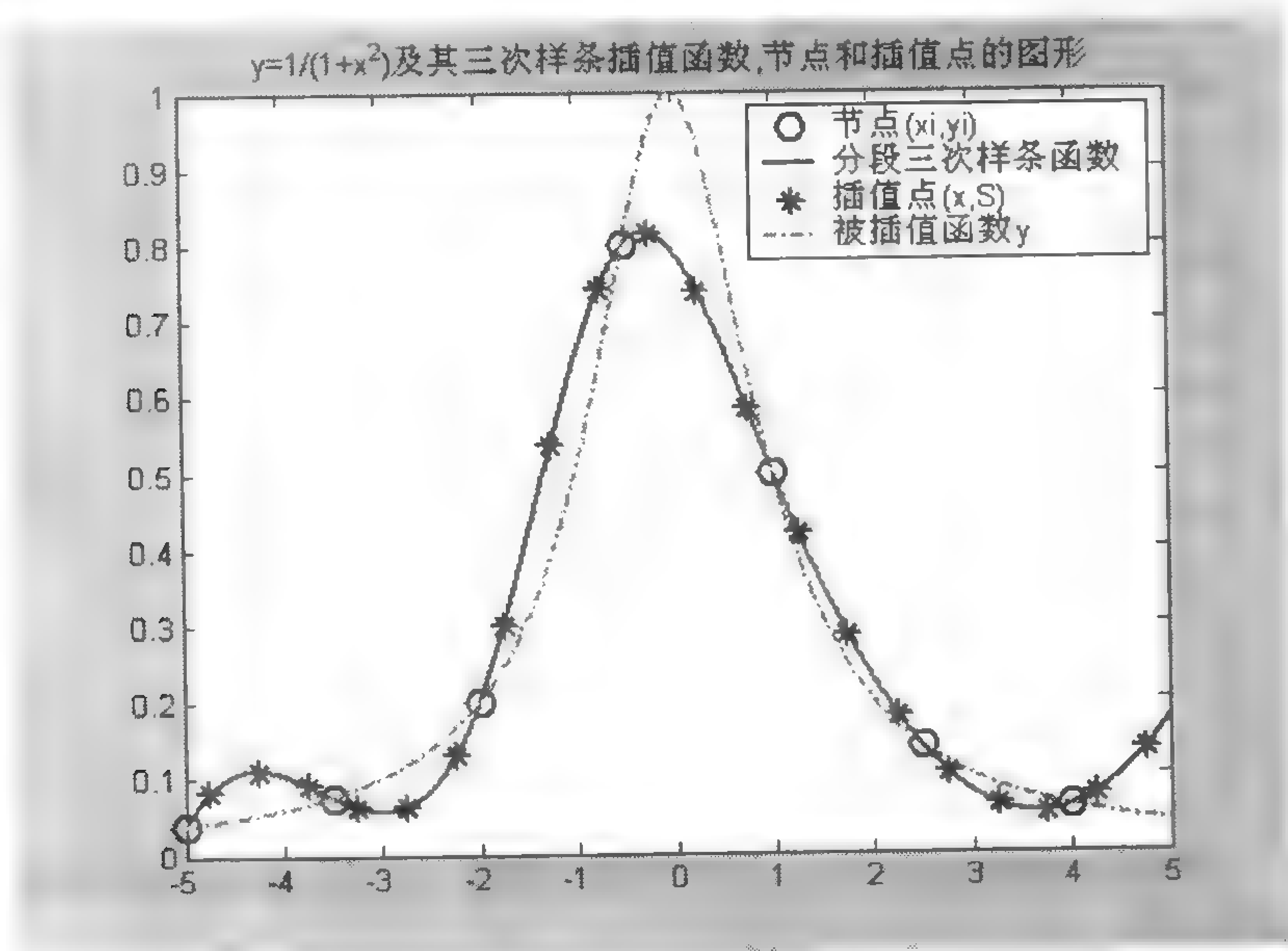


图 6－20 $f(x)=\frac{1}{1+x^2}$ 及其分段三次样条插值函数，节点和插值点的图形

（2）限定端点约束条件，取 $dy_1=dy_n=S_n'(\pm5)=0$，在 MATLAB 工作窗口输入程序

```
>> splinetx(x0,y0,x1,x,y,0,0)
title('y=1/(1+x^2)及其分段压紧三次样条函数,节点和插值点的图形')
```

运行后屏幕显示各小区间中点 $x_{i+\frac{1}{2}}$ 处 $S_n(x)$ 的值(略),作出节点,插值点,$f(x)$ 和 $S_n(x)$ 的图形(见图 6-21).

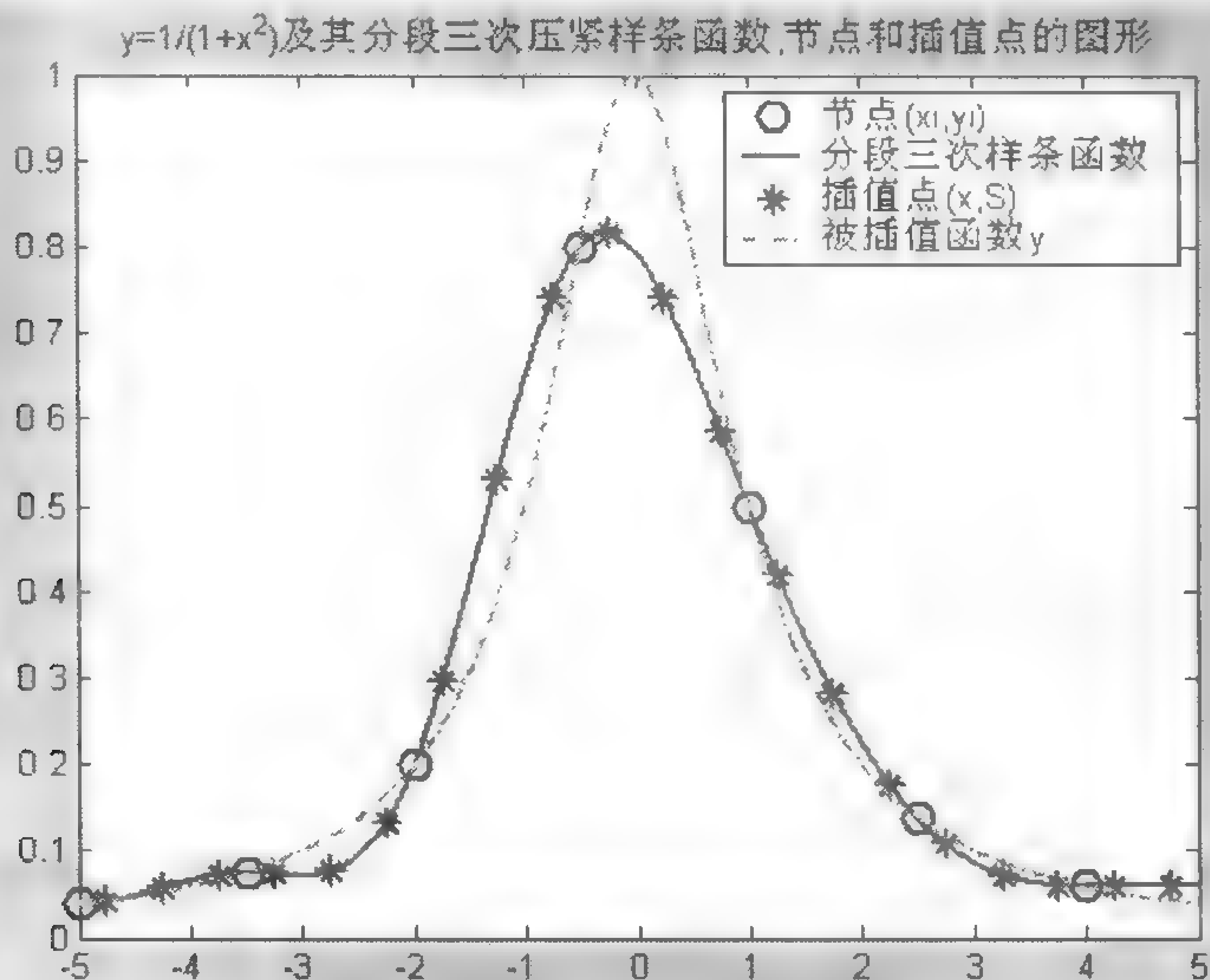


图 6-21 $f(x)=\dfrac{1}{1+x^2}$ 及分段三次压紧样条函数,节点和插值点的图形

从图 6-20 可见,在两端的区间[-5, -2]和[2.5,5]上,由于被插值曲线比较平缓,所以分段三次样条函数 $S_n(x)$ 图形比分段埃尔米特插值函数的图 6-14误差大. 但是如果调节端点约束条件(见压紧三次样条图 6-21)或者增加节点的倍数,例如取 $x_0=-5:0.5:5$,则三次样条函数与被插值函数的图像基本重合,见图 6-22.

例 6.7.10 设函数 $f(x)=0.5x-\cos x$ 定义在区间$[-2\pi,2\pi]$上,取 $n=7$,按等距节点构造分段三次样条函数 $S_n(x)$,用 MATLAB 程序计算各小区间中点 x_i 处 $S_n(x)$ 的值,分别作出局部和整体区间上的节点,插值点,$f(x)$,三次样条函数 $S_n(x)$ 和分段三次埃尔米特插值函数 $H_n(x)$ 的图形,并进行比较.

解 (1) 记节点的横坐标 $x_i=-2\pi+ih,h=4\pi/7,i=0,1,2,\cdots,7$,插值点 $x_{i+\frac{1}{2}}=\dfrac{1}{2}(x_i+x_{i+1}),i=0,1,2,\cdots,6$.

(2) 编写并保存名为 sanci.m 的 M 文件如下.

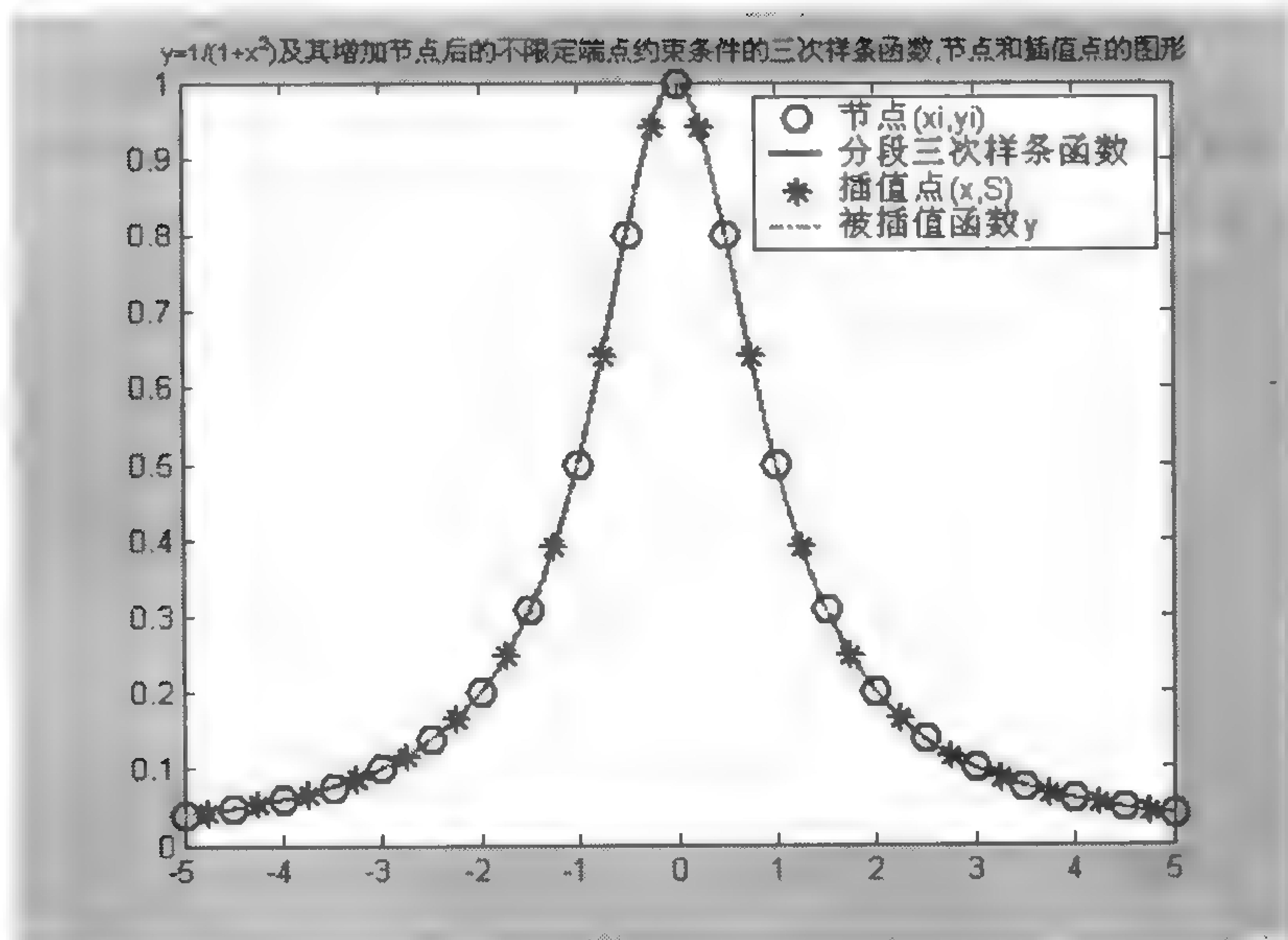


图 6-22 $f(x)=\dfrac{1}{1+x^2}$及其增加节点后的不限定端点约束条件的三次样条函数,节点和插值点的图形

```
subplot(2,2,1)
h = 4 * pi /7; x1 = -2 * pi:h: -2 * pi +3 * h;c1 =0.5. * x1 - cos(x1);
xx1 = -2 * pi + 4 * pi/14 :h: -2 * pi + pi/11 +2 * h; X1 = -2 * pi:
0.001: -2 * pi +3 * h;
Y1 =interp1(x1,c1,xx1,'pchip'); YY1 =interp1(x1,c1,X1,'pchip');
pp1 = interp1 (x1,c1,xx1,' spline '); P1 = interp1 (x1,c1,X1,'
spline ');
cc1 =0.5. * X1 - cos(X1);% pp1 = spline (x1,c1,xx1);
plot(x1,c1,'bo',xx1,pp1,'k *',X1,P1,'k -',xx1,Y1,'rx',X1,YY1,'r
:',X1,cc1,'g -.')
subplot(2,2,2)
x2 =2 * pi -3 * h:h:2 * pi;c2 =0.5. * x2 - cos(x2); xx2 =2 * pi -4 *
pi/14 -2 * h:h:2 * pi -4 * pi/14;
X2 =2 * pi -3 * h:0.001:2 * pi; pp2 =interp1 (x2,c2,xx2,'spline');
YY2 =interp1(x2,c2, X2,'pchip'); Y2 =interp1(x2,c2,xx2,'pchip');
P2 =interp1 (x2,c2,X2,' spline '); cc2 =0.5. * X2 - cos(X2);
plot(x2,c2,'bo',xx2,pp2,'k *',X2,P2,'k -',xx2,Y2,'rx',X2,YY2,'r:',
```

```
X2,cc2,'g-.')
        subplot(2,1,2)
        x=-2*pi:h:2*pi;c=0.5.*x-cos(x); xx=-2*pi+4*pi/14:h:
2*pi-4*pi/14;
        pp=spline(x,c,xx),Y=interp1(x,c,xx,'pchip'), X=-2*pi:
0.001:2*pi;
        P=interp1 (x,c,X,'spline'); YY=interp1(x,c,X,'pchip'); cc=
0.5.*X-cos(X);
        plot(x,c,'bo',xx,pp,'k*',X,P,'k-',xx,Y,'rx',X,YY,'r:',X,cc,'g-.')
        legend('节点(xi,yi)','三次样条插值','三次样条插值函数','分段三次埃尔米
特插值','分段三次埃尔米特插值函数','y=0.5x-cosx 的函数')
        title('y=0.5x-cos(x)及其三次样条函数,分段三次埃尔米特插值函数,节
点和插值点的图形')
```

(3) 在 MATLAB 工作窗口输入文件名

```
>> sanci
```

运行后屏幕显示各小区间中点 x_i 处三次样条插值 pp 和分段三次埃尔米特插值 Y,出现被插值函数 $f(x)$,节点,三次样条和分段三次埃尔米特的函数及其插值点等图形(见图 6-23)如下

```
        pp =
           -3.2181   -0.9609   -0.6824   -0.9476   1.1128   2.6295
2.1675
        Y =
           -3.0085   -1.0074   -0.7589   -0.7872   1.1498   2.4072
2.3787
```

由图 6-23 可见,总体上,三次样条比分段埃尔米特插值误差小.

6.7.8 用 MATLAB 计算有关分段三次样条的误差

利用分段埃尔米特插值函数可以推导出下面三次样条函数的误差估计定理.

定理 6.14 如果被插值函数 $f(x)$ 在插值区间 $[a,b]$ 上具有 4 阶连续导数, $[a,b]$ 上的 $n+1$ 个节点 (x_i,y_i) $(i=0,1,2,\cdots,n)$ 满足 $a=x_0<x_1<x_2<\cdots<x_n=b$, $S_n(x)$ 是定义 6.5 定义的 $y=f(x)$ 在点 (x_i,y_i) $(i=0,1,2,\cdots,n)$ 处的三次样条函数,则

(1) $S_n(x)$ 在插值区间 $[a,b]$ 上的误差公式可以表示为

$$|R_n(x)|=|f(x)-S_n(x)|\leqslant\frac{1}{384}h^4\|f^{(4)}(x)\|_\infty, \tag{6.94}$$

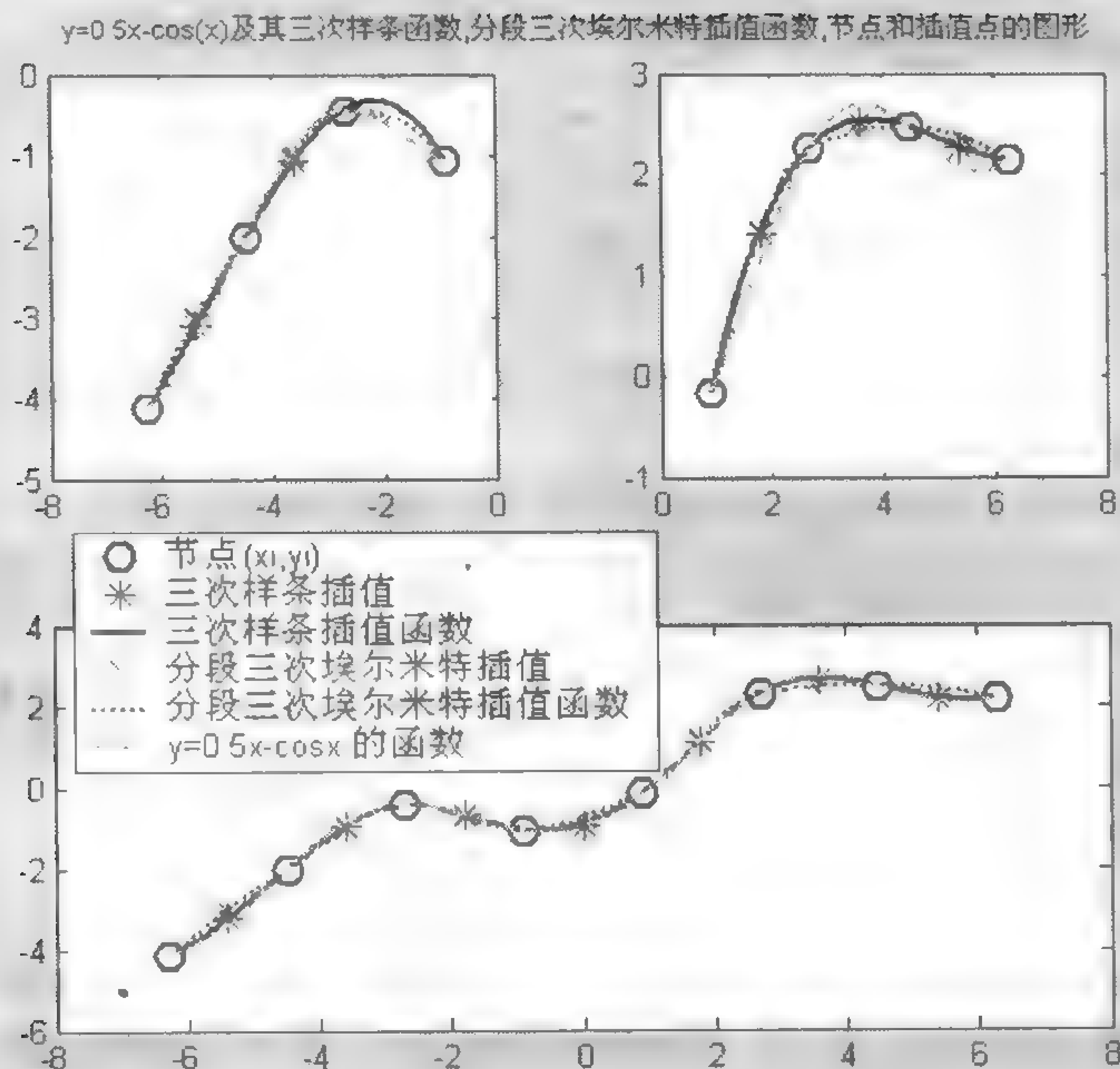


图 6－23 $y=0.5x-\cos x$ 及其三次样条函数，分段三次埃尔米特插值函数，节点和插值点的图形

其中 $h=\max\limits_{1\leqslant i\leqslant n}|x_i-x_{i-1}|$.

(2)
$$\lim_{n\to\infty}S_n(x)=f(x). \tag{6.95}$$

且 $S_n(x)$ 在 $[a,b]$ 上一致收敛到 $f(x)$. (6.95)式表明像分段线性函数一样，三次样条函数 $S_n(x)$ 也具有良好的收敛性. 实际上，只要被插值函数 $f(x)$ 在插值区间 $[a,b]$ 上具有一阶连续导数，就有

$$|R_n(x)|=|f(x)-S_n(x)|\leqslant\frac{9}{8}h\max_{|x_1-x_2|\leqslant h}|f(x_1)-f(x_2)|\to 0\ (h\to 0). \tag{6.96}$$

例 6.7.11 如果函数 $f(x)=\dfrac{1}{1+25x^2}$ 定义在区间 $[-1,1]$ 上，取 $n=10$，按等距节点构造分段三次样条函数 $S_n(x)$，用 MATLAB 程序计算在 $[-1,1]$ 上 $\max\limits_{-1\leqslant x\leqslant 1}|f^{(4)}(x)|$ 和 $S_n(x)$ 的含 $h=|x_i-x_{i-1}|$ 误差公式和误差限.

解 (1) $S_n(x)$ 在区间 $[-1,1]$ 上的误差公式可以表示为

$$|R_{10}(x)| = |f(x) - S_{10}(x)| \leqslant \frac{1}{384}h^4 \| f^{(4)}(x) \|_\infty,$$ 其中 $h=0.2$.

(2) 在 MATLAB 工作窗口输入程序

```
>> syms  x,y =1/(1 +25 * x^2); yxxxx = diff(y,x,4),
```

运行后输出 $f(x)$ 的 4 阶导数.

(3) 在 MATLAB 工作窗口输入程序

```
>> syms h,x = -1:0.0001:1;
   yxxxx =150000000./(1 +25.*x.^2).^5.*x.^4 -4500000./(1 +25.*
x.^2).^4.*x.^2 +15000./(1 +25.*x.^2).^3;
   myxxxx = norm(yxxxx,inf), R =(h^4) * myxxxx/384
```

运行后输出 $f(x)$ 的 4 阶导数在区间 $[-1,1]$ 上的最大值 *myxxxx* 和 $H_{n,3}(x)$ 在区间 $[-1,1]$ 上的误差公式为

```
myxxxx =                R =
          15000            625/16 * h^4
```

(4) 在 MATLAB 工作窗口输入程序

```
>> h =0.2; R =625/16 * h^4
```

运行后输出误差限为

```
R =
  0.06250000000000
```

例 6.7.12 设函数 $f(x) = \tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$ 定义在区间 $[-\pi,\pi]$ 上,取 $n=7$,按等距节点分别作分段线性插值、拉格朗日插值、三次样条插值和分段埃尔米特插值. 用 MATLAB 程序计算各小区间中点 x_i 处四种插值及其绝对误差和相对误差,作出 $f(x)$ 及其后三种插值函数,插值点和节点的图形,并进行比较.

解 (1) 记节点的横坐标 $x_i = -\pi + ih, h = 2\pi/7, i = 0,1,2,\cdots,7$,插值点 $x_{i+\frac{1}{2}} = \frac{1}{2}(x_i + x_{i+1}), i=0,1,2,\cdots,6.$

(2) 编写并保存名为 sanli679.m 的 M 文件如下

```
h =2 * pi /n; x =( -2 * pi +h)/2:h:(2 * pi -h)/2;
y =tan(cos((3^(1/2) +sin(2 * x))./(3 +4 * x.^2)));
x0 = -pi:h:pi;X = -pi:h/12:pi;
y0 =tan(cos((3^(1/2) +sin(2 * x0))./(3 +4 * x0.^2)));
Y =tan(cos((3^(1/2) +sin(2 * X))./(3 +4 * X.^2)));
YL =lagr1(x0,y0,X); YS =interp1(x0,y0,X,'spline');
YH =interp1(x0,y0,X, 'pchip'); yL =lagr1(x0,y0,x);
yX =interp1(x0,y0,x); yS =interp1(x0,y0,x,'spline');
```

```
yH = interp1(x0,y0,x,'pchip'); RL = abs((y - yL)./y);
RS = abs((y - yS)./y);
RH = abs((y - yH)./y); RX = abs((y - yX)./y);RLj = abs(y - yL);
mRLj = mean(RLj);
RSj = abs(y - yS);
mRSj = mean(RSj);RHj = abs(y - yH);RXj = abs(y - yX);
mRHj = mean(RHj);
mRXj = mean(RXj);mRL = mean(RL);mRX = mean(RX);
mRS = mean(RS);mRH = mean(RH);
CZ = [x' y' yL' yX' yS' yH'],R = [x' RL' RX' RS' RH'],
mR = [mRL' mRX' mRS' mRH']
Rj = [x' RLj' RXj' RSj' RHj'],mRj = [mRLj' mRXj' mRSj' mRHj'],
plot(x0,y0,'bo',x,yS,'r*',X,Y,'k-',X,YL,'g-.',X,YS,'c:', X,YH,'m--'),
legend('节点','三次样条插值点','被插值函数','拉格朗日插值函数','三次样条函数','分段三次埃尔米特插值函数')
title('y = tan(cos((sqrt(3) + sin(2x))/(3 + 4x^2)))及其三种插值函数,节点和插值点的图形')
```

(3) 运行程序

```
>> n = 7;sanli679
```

得到各小区间中点 $x_{i+\frac{1}{2}}$ 处的函数值 $y(x_{i+\frac{1}{2}})$，分段线性插值 $X_n(x_{i+\frac{1}{2}})$、拉格朗日插值 $L_n(x_{i+\frac{1}{2}})$、三次样条插值 $S_n(x_{i+\frac{1}{2}})$ 和分段埃尔米特插值 $H_n(x_{i+\frac{1}{2}})$ 及其绝对误差、相对误差和平均值的结果如表 6－12、表 6－13 和表 6－14. 作出 $f(x)$ 及其后三种插值函数，插值点和节点的图形如图 6－24 所示.

表 6－12 $f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$ 在各小区间中点 $x_{i+\frac{1}{2}}$ 处的函数值和四种插值

序号 i	小区间中点 $x_{i+\frac{1}{2}}$	函数值 $y(x_{i+\frac{1}{2}})$	拉格朗日插值 $L_n(x_{i+\frac{1}{2}})$	分段线性插值 $X_n(x_{i+\frac{1}{2}})$	三次样条插值 $S_n(x_{i+\frac{1}{2}})$	分段埃尔米特插值 $H_n(x_{i+\frac{1}{2}})$
0	-2.692 8	1.546 9	1.953 0	1.544 4	1.551 4	1.541 7
1	-1.795 2	1.526 1	1.404 2	1.532 3	1.521 2	1.532 4
2	-0.897 6	1.532 4	1.602 1	1.493 2	1.550 7	1.508 2
3	0	1.111 0	1.179 9	1.232 3	1.179 7	1.216 4
4	0.897 6	1.278 1	1.125 4	1.246 2	1.192 8	1.230 6
5	1.795 2	1.546 0	1.760 4	1.519 7	1.590 9	1.535 3
6	2.692 8	1.555 9	0.953 3	1.555 1	1.502 0	1.555 4

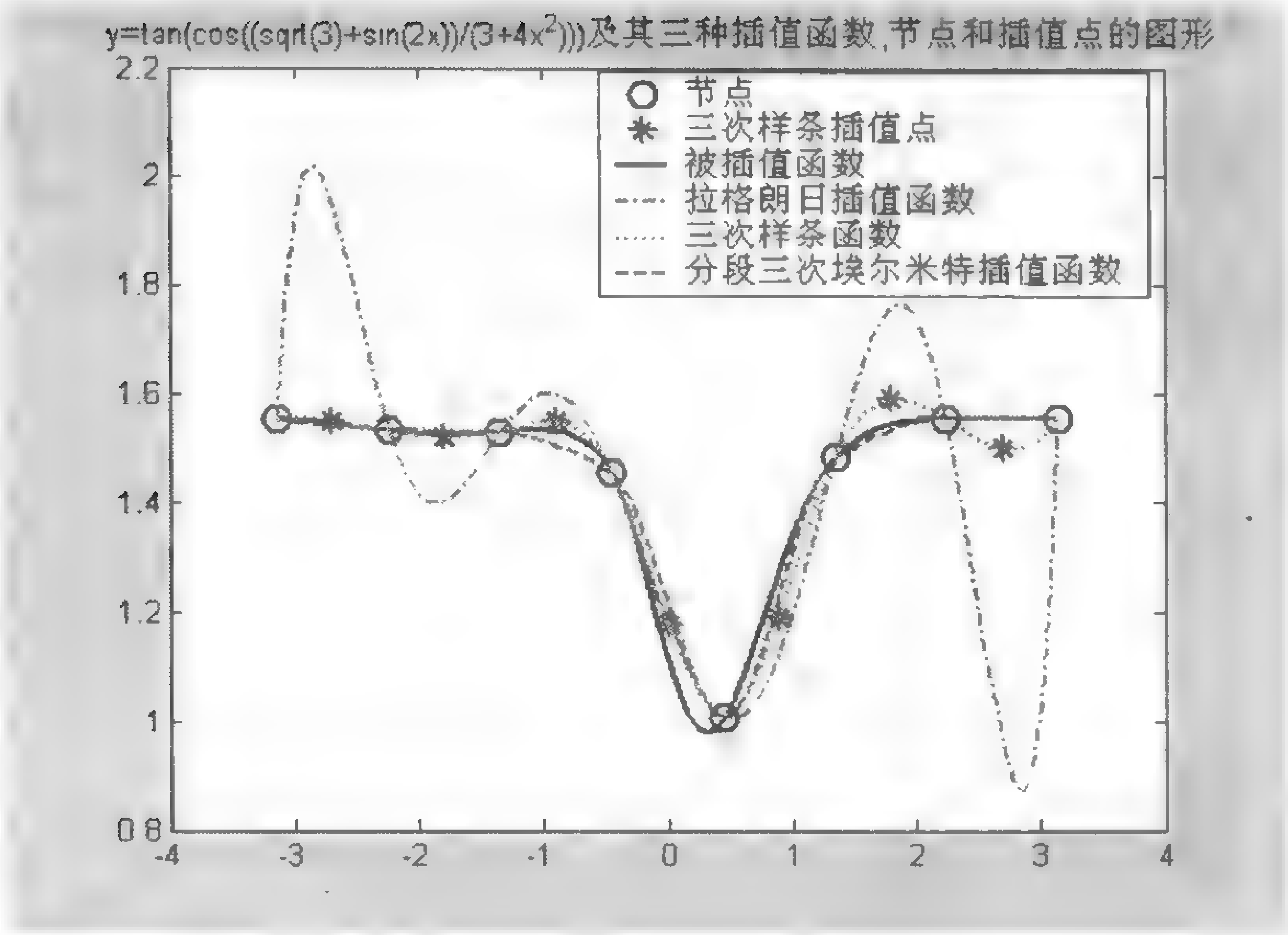


图 6－24 $f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$及其三种插值函数,节点和插值点的图形

表 6－13 $f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$**在各小区间中点** $x_{i+\frac{1}{2}}$ **处的四种插值的相对误差**

序号 i	小区间中点 $x_{i+\frac{1}{2}}$	拉格朗日插值相对误差 $RL_n(x_{i+\frac{1}{2}})$	分段线性插值相对误差 $RX_n(x_{i+\frac{1}{2}})$	三次样条插值相对误差 $RS_n(x_{i+\frac{1}{2}})$	分段埃尔米特插值相对误差 $RH_n(x_{i+\frac{1}{2}})$
0	－2.692 8	0.262 5	0.001 6	0.002 9	0.003 4
1	－1.795 2	0.079 9	0.004 1	0.003 2	0.004 1
2	－0.897 6	0.045 5	0.025 6	0.011 9	0.015 8
3	0	0.062 1	0.109 2	0.061 9	0.094 9
4	0.897 6	0.119 5	0.025 0	0.066 7	0.037 2
5	1.795 2	0.138 6	0.017 1	0.029 0	0.007 0
6	2.692 8	0.387 3	0.000 5	0.034 6	0.000 3
平均值	0	0.156 5	0.026 1	0.040 1	0.023 2

表 6-14 $f(x)=\tan\left(\cos\left(\frac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$在各小区间中点 $x_{i+\frac{1}{2}}$ 处的四种插值的绝对误差

序号 i	小区间中点 $x_{i+\frac{1}{2}}$	拉格朗日插值绝对误差 $RL_n(x_{i+\frac{1}{2}})$	分段线性插值绝对误差 $RX_n(x_{i+\frac{1}{2}})$	三次样条插值绝对误差 $RS_n(x_{i+\frac{1}{2}})$	分段埃尔米特插值绝对误差 $RH_n(x_{i+\frac{1}{2}})$
0	-2.692 8	0.406 1	0.002 5	0.004 5	0.005 2
1	-1.795 2	0.121 9	0.006 2	0.004 9	0.006 3
2	-0.897 6	0.069 7	0.039 2	0.018 3	0.024 2
3	0	0.069 0	0.121 3	0.068 8	0.105 4
4	0.897 6	0.152 7	0.031 9	0.085 3	0.047 5
5	1.795 2	0.214 4	0.026 4	0.044 9	0.010 8
6	2.692 8	0.602 6	0.000 8	0.053 9	0.000 4
平均值	0	0.233 8	0.032 6	0.040 1	0.028 6

将表 6-12 中的四种插值结果与精确值 $y(x_{i+\frac{1}{2}})$ 相比较，从表 6-13 中的四种插值的相对误差，表 6-14 中的四种插值的绝对误差和图 6-24，显然它们在节点处相等，而在插值点处分段埃尔米特插值的结果最好，拉格朗日插值在区间两端出现振荡，其误差最大.

例 6.7.13（机床加工） 待加工零件的外形根据工艺要求由一组数据 (x,y) 给出（在平面情况下），用程控铣床加工时每一刀只能沿 x 方向和 y 方向走非常小的一步，这就需要从已知数据得到加工所要求的步长很小的 (x,y) 坐标. 表 6-15给出的 x,y 数据位于机翼断面的下轮廓线上（如图 6-25 所示），假设需要得到 x 坐标每改变 0.1 时的 y 坐标. 试完成加工所需数据，画出曲线，并求出 $x=0$ 处的曲线斜率和 $13\leqslant x\leqslant 15$ 范围内 y 的最小值.

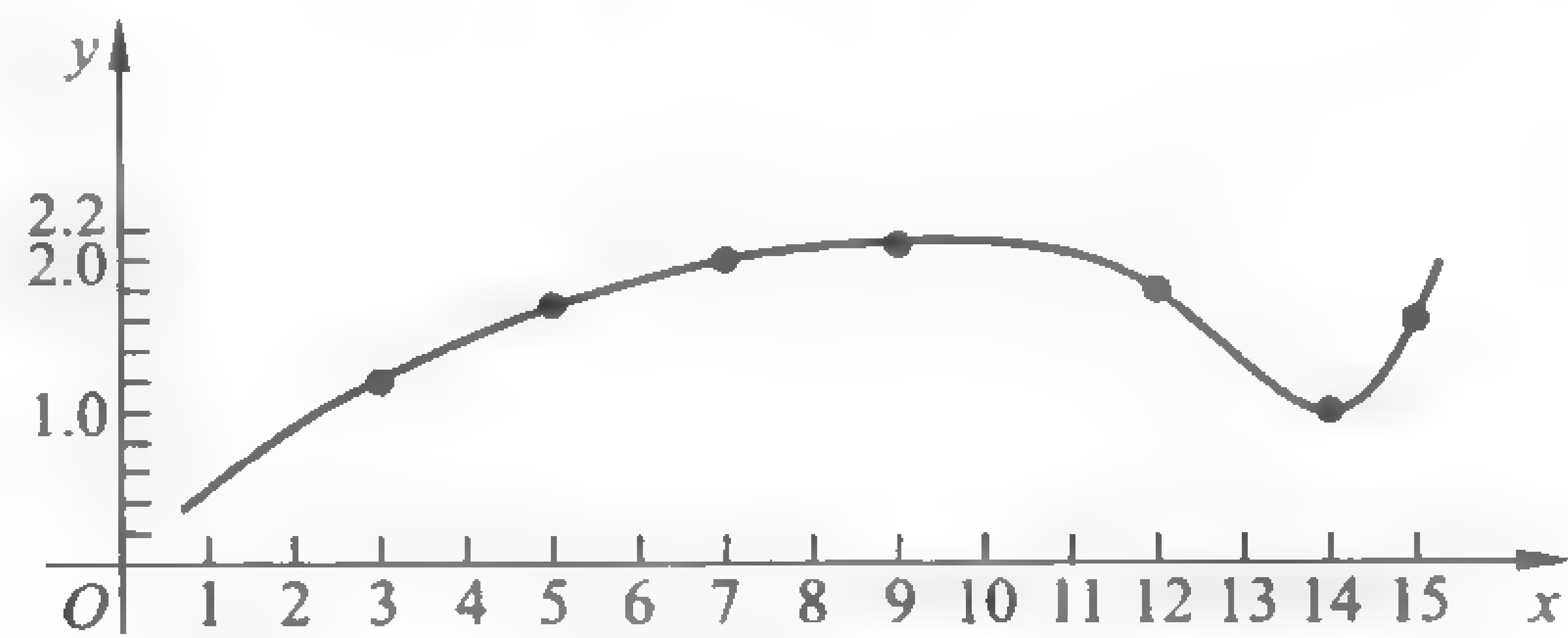


图 6-25 机翼断面轮廓线（表 6-15 数据用圆点表示）

表 6－15 机翼断面下轮廓线上的部分数据

X	0	3	5	7	9	11	12	13	14	15
Y	0	1.2	1.7	2.0	2.1	2.0	1.8	1.2	1.0	1.6

解 根据上述提出的加工要求，以所给数据为节点，在 $x=0$ 到 $x=15$ 范围内求步长为 0.1 的插值. 用四种插值方法试验，编写并保存名为 sancili6710.m 程序为 M 文件.

```
x0 =[0 3 5 7 9 11 12 13 14 15 ]; x =0:0.1:15;
y0 =[0 1.2 1.7 2.0 2.1 2.0 1.8 1.2 1.0 1.6 ]; yL - lagr1(x0,y0,x);
yX = interp1(x0,y0,x); yS = interp1(x0,y0,x,'spline');
yH = interp1(x0,y0,x, 'pchip'); CZ =[x' yL' yX' yS' yH']
subplot(4,1,1)
plot(x0,y0,'bo',x,yL,'r'), grid,title('拉格朗日插值')
subplot(4,1,2)
plot(x0,y0,'bo',x,yX,'r'), grid,title('分段线性插值')
subplot(4,1,3)
plot(x0,y0,'bo',x,yS,'r'), grid,title('三次样条')
subplot(4,1,4)
plot(x0,y0,'bo',x,yH,'r'), grid,title('分段埃尔米特插值')
```

在 MATLAB 工作窗口输入文件名

```
>> sancili6710
```

运行后得到的拉格朗日插值、分段线性插值、三次样条插值和分段埃尔米特插值及其节点的图形（见图 6－26），同时还得到拉格朗日插值、分段埃尔米特插值、分段线性插值和三次样条插值的结果（见表 6－16）.

表 6－16 机床加工四种插值结果

插值点 x_i	拉格朗日插值 $L_n(x_i)$	分段线性插值 $X_n(x_i)$	三次样条插值 $S_n(x_i)$	分段埃尔米特插值 $H_n(x_i)$
0	0	0	0	0
0.100 0	−4.943 7	0.040 0	0.049 9	0.048 7
0.200 0	−8.820 0	0.080 0	0.099 0	0.096 9
0.300 0	−11.772 6	0.120 0	0.147 4	0.144 5
⋮	⋮	⋮	⋮	⋮
13.800 0	0.945 4	1.040 00	0.982 8	1.011 2
⋮	⋮	⋮	⋮	⋮
14.800 0	1.570 5	1.480 0	1.405 7	1.409 6
14.900 0	1.605 0	1.540 0	1.497 9	1.502 2
15.000 0	1.600 0	1.600 0	1.600 0	1.600 0

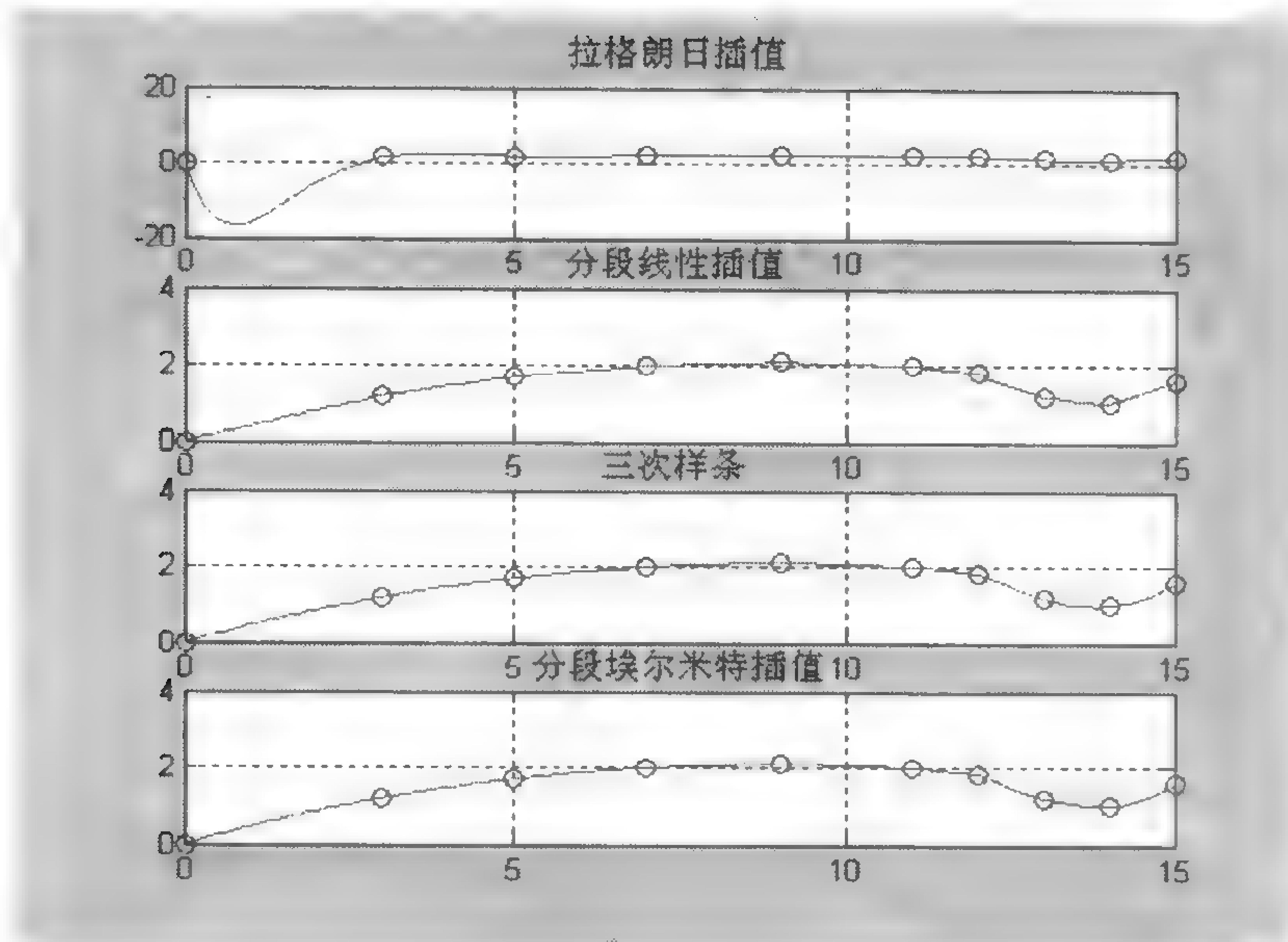


图 6-26 机床加工四种插值结果的图形

可以看出,拉格朗日插值的结果根本不能应用,分段线性插值的光滑性较差(特别是在 $x=14$ 附近弯曲处),建议选用三次样条插值或分段埃尔米特插值的结果.

拉格朗日插值是高次多项式插值($n+1$ 个节点上用不超过 n 次的多项式),插值曲线光滑,误差估计有表达式.但有振荡现象,收敛性不能保证.这种插值主要用于理论分析,实际意义不大.

分段线性插值是低次多项式插值,简单实用,收敛性有保证,但不光滑.三次样条插值或分段埃尔米特插值的整体光滑性已大有提高,应用广泛,但是误差估计较困难.



习 题 6.7

1. 作函数 $f(x)=\dfrac{1}{1+25x^2}$ 在区间 $[-5,5]$ 上的四种插值函数(包括三次样条)的图形($n=2,4,6,8,10$),然后进行比较,并讨论 n 的大小与误差 $|R_n(x)|$ 的关系.

2. 设函数 $f(x)=\sin\dfrac{3-\cos 4x}{1+25x^2}$ 定义在区间 $[-\pi,\pi]$ 上,取 $n=13$,按等距节点求四种插值

函数(包括三次样条),并用 MATLAB 程序计算各小区间中点处 $S_n(x)$ 的值及其相对误差,并进行比较.

3. 设函数 $f(x)=\dfrac{1}{1+x^2}$ 定义在区间 $[-5,5]$ 上,取 $n=10$,按等距节点构造四种插值函数,用 MATLAB 程序计算各小区间中点 x_i 处的值,作出节点、插值点、$f(x)$ 和四种插值函数的图形.

4. 设函数 $f(x)=0.15x^2-\sin(2x-1)$ 定义在区间 $[-\pi,\pi]$ 上,取 $n=7$,按等距节点构造四种插值函数,用 MATLAB 程序计算各小区间中点 x_i 处插值,作出节点、插值点、$f(x)$ 和四种插值函数的图形.

5. 设函数 $f(x)=\tan\left(\cos\left(\dfrac{\sqrt{3}+\sin 2x}{3+4x^2}\right)\right)$ 定义在区间 $[-1,1]$ 上,取 $n=10$,按等距节点构造四种插值函数 $S_n(x)$.

(1) 用 MATLAB 程序计算各小区间中点 x_i 处 $S_n(x)$ 的值,作出节点、插值点、$f(x)$ 和四种插值函数的图形;

(2) 并用 MATLAB 程序计算各小区间中点处四种插值及其相对误差;

(3) 用 MATLAB 程序估计 $\max\limits_{-\pi\leqslant x\leqslant\pi}|f''(x)|$ 和 $S_n(x)$ 在区间 $[-\pi,\pi]$ 上的误差限.

6. 选择一些函数,在 n 个节点上(n 不要太大,如 5 ~ 11)用拉格朗日、分段线性、三次样条、分段埃尔米特插值四种插值方法,计算 m 个插值点的函数值(m 要适中,如 50 ~ 100). 通过数值和图形输出,将三种插值结果与精确值进行比较. 适当增加 n,再作比较,由此作初步分析. 下列函数供选择参考:

(1) $y=\sin x,\quad 0\leqslant x\leqslant 2\pi$;

(2) $y=\sqrt{1-x^2},\ -1\leqslant x\leqslant 1$;

(3) $y=\cos^{10}x,\ -2\leqslant x\leqslant 2$;

(4) $y=\exp(-x^2),\ -2\leqslant x\leqslant 2$.

7. 用 $y=\sqrt{x}$ 在 $x=0,1,4,9,16$ 产生 5 个节点 $P_1,\cdots,P_5$. 用不同的节点构造插值公式来计算 $x=5$ 处的插值(如用 $P_1\cdots P_5$;$P_1\cdots P_4$;$P_2\cdots P_4$ 等),与精确值比较并进行分析.

6.8 高元插值及其 MATLAB 程序

本章前面的各节讲述的都是一元插值,即节点都是一元变量,插值函数是一元函数(曲线). 如果节点都是二元变量,插值函数是二元函数(曲面),则可以构造二元插值. 例如,在某区域测量了若干点(节点)的高程(节点值),为了画出较精确的等高线图,就要先插入更多的点(插值点),计算这些点的高程(插值). 如果节点都是三元变量,插值函数是三元函数,则可以构造三元插值. 二元插值及二元插值以上的插值称为高元插值. 下面主要介绍二元插值和三元插值及其 MATLAB 程序.

6.8.1 meshgrid 命令的功能和调用格式

常用的二元插值方法有最近邻插值(nearest neighbor interpolation)、双线性插值(bilinear interpolation)、双三次插值(bicubic interpolation)和样条插值(spline interpolation). 常用的三元插值方法有最近邻插值(nearest neighbor interpolation)、线性插值(linear interpolation)、三次插值(cubic interpolation)和样条插值(spline interpolation),也用于拟合方法. 这些方法在 MATLAB 函数库中有编好的名为 interp2 .m 和 interp3 .m 的计算程序,该程序有几种调用格式.计算时,首先根据网格坐标命令 meshgrid 把节点坐标的常数向量 x 和 y 或 z 转化为矩阵 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 或 $\boldsymbol{Z}$. 这些所有的插值方法都要求利用名为 meshgrid 的程序产生矩阵,然后选用对应的调用格式直接用 interp2 或 interp3 命令计算即可.$\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 或 $\boldsymbol{Z}$ 可以是不等距分布. 不但我们用 interp2 或 interp3 命令计算二元插值或三元插值时需要用 meshgrid 命令产生矩阵 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 或 $\boldsymbol{Z}$,而且作三元图形也需要用 meshgrid 命令产生矩阵.所以,首先介绍 meshgrid 的 M 函数文件的功能、详细的调用方法.

常用的 meshgrid 命令的调用格式有三种,分别介绍如下:

调用格式一 [X,Y] = meshgrid (x,y)

[X,Y] = meshgrid (x,y)将向量 x 和 y 转换成矩阵 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$,其中矩阵 $\boldsymbol{X}$ 的每行是向量 x,矩阵 $\boldsymbol{Y}$ 的每列是向量 y,此命令可以被用于计算二元函数或作二元曲面的图形. 向量 x 和 y 的输入形式有两种.

例如,输入程序

```
>> x = [1,3,4,11,7];y = [4, -5,2, -9];[X,Y] = meshgrid(x,y)
```

运行后屏幕显示

```
X =                              Y =
   1   3   4   11   7               4    4    4    4    4
   1   3   4   11   7              -5   -5   -5   -5    5
   1   3   4   11   7               2    2    2    2    2
   1   3   4   11   7              -9   -9   -9   -9   -9
```

请读者输入

```
>> [X,Y] = meshgrid( -2:1.5:2, -2:1.4:2)
```

观察运行后屏幕显示结果.

例 6.8.1 已知 $x=-3:0.2:3$;$y=x$,计算函数 $z=7-3x^4e^{-x^2-y^2}$ 的值,并作出函数的图形.

解 输入程序

```
>> [X,Y] = meshgrid(-3:.2:3, -3:.2:3);
```

```
Z=7-3* X.^4 .* exp(-X.^2 - Y.^2), mesh(Z)
title('Z=7-3 X^4 exp(-X^2-Y^2)的图形')
```

运行后输出函数值(略)和图形如图 6-27.

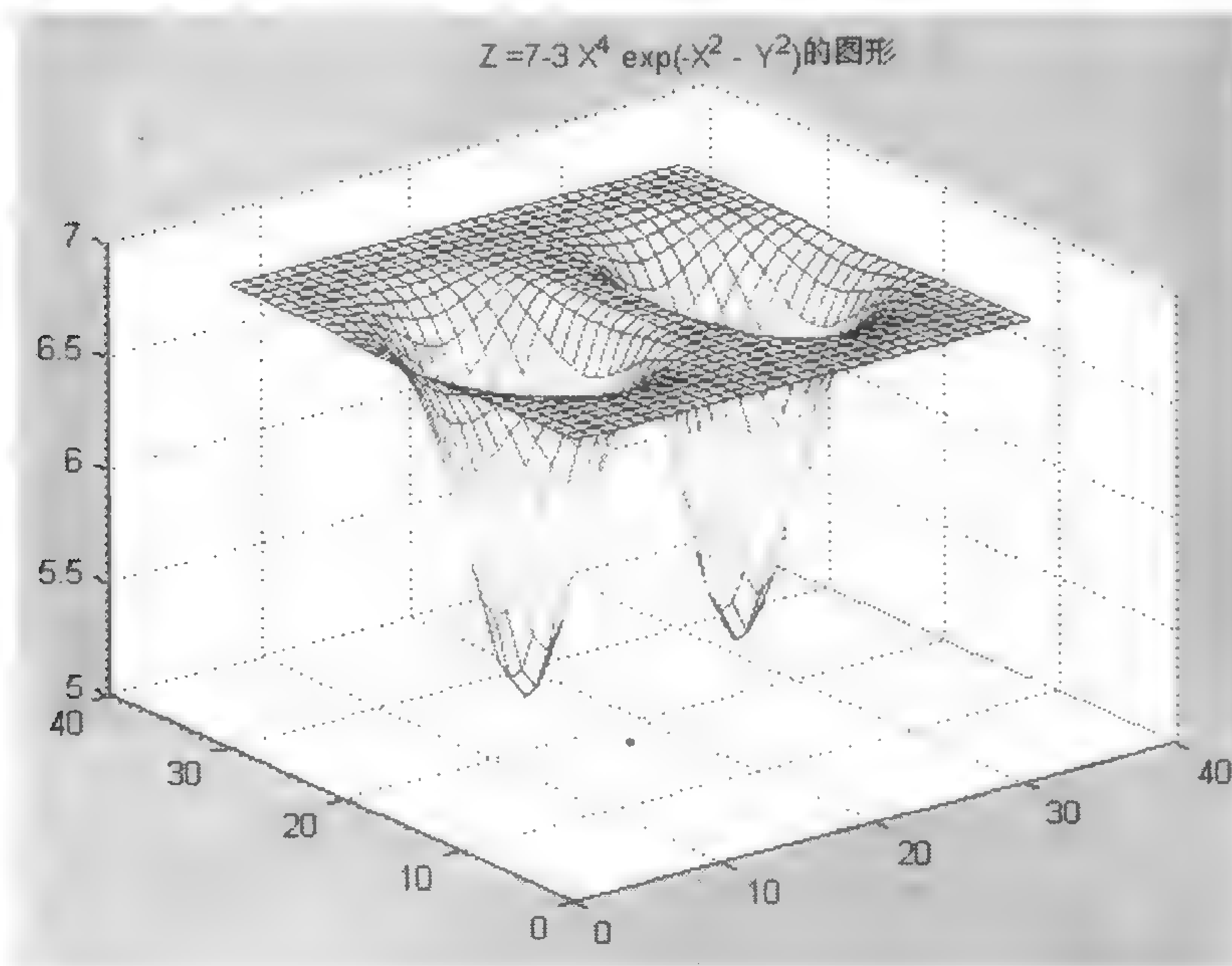


图 6-27 函数 $z=7-3x^4e^{-x^2-y^2}$ 的图形

例 6.8.2 作出函数 $z=2+xe^{-x^2-y^2}$ 在区域 $-2\leqslant x\leqslant 2$, $-2\leqslant y\leqslant 2$ 上的图形.

解 输入程序

```
>> [X,Y]=meshgrid(-2:.2:2, -2:.2:2);Z=2+X .* exp(-X.^2 -
Y.^2); mesh(Z)
title('Z=2+X exp(-X^2 - Y^2)的图形')
```

运行后输出函数值(略)和图形如图 6-28 所示.

调用格式二 [X,Y]=meshgrid (x)

是[X,Y]=meshgrid(x,x)的一种缩写式.

调用格式三 [X,Y,Z]=meshgrid (x,y,z)

此命令将向量 $\boldsymbol{x}$,$\boldsymbol{y}$ 和 $\boldsymbol{z}$ 转换成矩阵 $\boldsymbol{X}$,$\boldsymbol{Y}$ 和 $\boldsymbol{Z}$,经常被用于计算三元函数插值或作三元立体的图形.

例 6.8.3 计算函数 $U=2+xe^{-x^2-y^2-z^2}$ 在 $x=0,1,2,-4$,$y=0,-1,3,5$,$z=y$ 处的函数值.

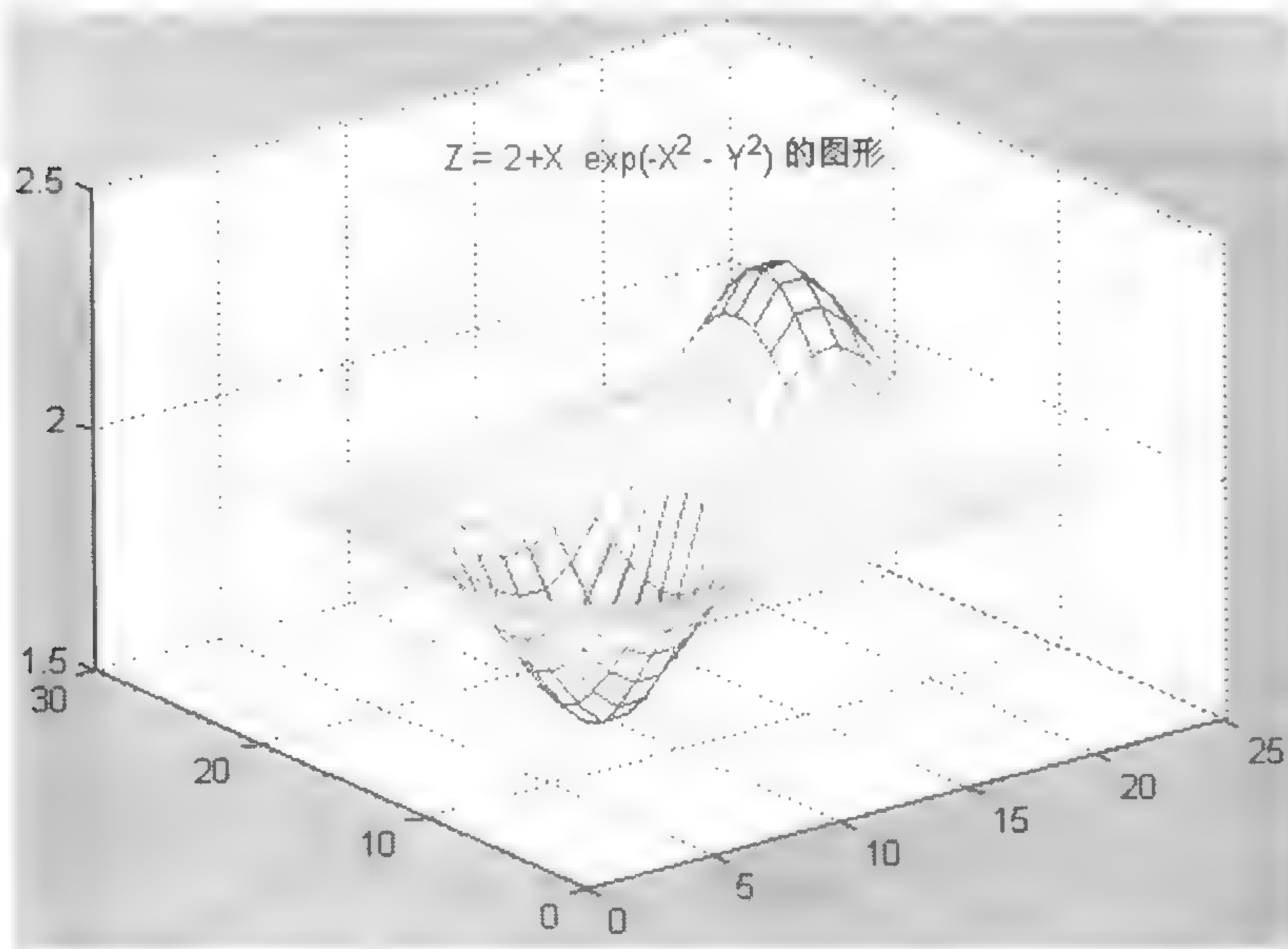


图 6-28 函数 $z=2+xe^{-x^2-y^2}$ 的图形

解 输入程序

```
>> x =[0,1,2, -4];y =[0, -1,3,5];z =y; [X,Y,Z] =meshgrid(x,y,z),
U =2 +X .* exp( -X.^2 - Y.^2 - Z.^2)
```

运行后屏幕显示略.

6.8.2 单调数据点上的二元插值及其 MATLAB 程序

二元插值函数在图像处理和数据可视化领域有广泛的应用.一元函数的插值概念可以类似地推广到二元函数的情形,只是数据点($\boldsymbol{x},\boldsymbol{y},z$)给出的形式不同,其中 $\boldsymbol{x}=(x_1,x_2,\cdots,x_m)$ 是行向量,$\boldsymbol{y}=(y_1,y_2,\cdots,y_n)^{\mathrm{T}}$ 是列向量,$\boldsymbol{z}$ 是 $m\times n$ 阶矩阵,即

$$z=\begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \vdots & \vdots & & \vdots \\ z_{m1} & z_{m2} & \cdots & z_{mn} \end{pmatrix}.$$

二元函数插值的主要目的是:在区域 $D=[x_1,x_m]\times[y_1,y_n]$ 上或者在 D 的分片子域 D_{ij} 上,构造被插值函数 $z=f(x,y)$ 的曲面上的点列 $\{(x_i,y_i,z_i)\}_{i=0}^{n}$ 的较简单函数 $z=P(x,y)$(例如多项式函数),使二元插值函数 $P(x,y)$ 逼近被插值函数 $f(x,y)$.

在 MATLAB 系统中，为我们提供了计算单调数据点上的二元插值程序 interp2，计算时直接调用.这里的**单调数据点** $\{(x_i,y_i,z_i)\}_{i=0}^{n}$ 是指 $\{x_i\}_{i=0}^{n}$ 和 $\{y_i\}_{i=0}^{n}$ 必须是单调序列（单调增加或单调减少）.程序 interp2 是计算常用的二元插值方法中最近邻插值、双线性插值、双三次插值和样条插值的 MATLAB 程序的文件名.计算时输入的节点 $(\boldsymbol{x},\boldsymbol{y})$ 中的向量 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$，$\boldsymbol{y}=(y_1,y_2,\cdots,y_n)$ 的元素必须分别是单调排列，然后用网格化命令[X,Y] = meshgrid(x,y)将 $\boldsymbol{x},\boldsymbol{y}$ 化为二元网格坐标 $\boldsymbol{X},\boldsymbol{Y}$，最后调用 interp2 程序.该程序有几种调用格式如表 6-17.

表 6-17 单调数据点上的二元插值的 MATLAB 命令

二元插值法的命令 interp2	功 能
ZI = interp2(X,Y,Z,XI,YI)	ZI = interp2(X,Y,Z,XI,YI)命令的主要功能是计算二元函数 Z 在插值点 $(\boldsymbol{X}_i,\boldsymbol{Y}_i)$ 处的二元线性插值所对应的向量 $\boldsymbol{Z}_i$. 输入的节点矩阵 $(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z})$ 中的 $\boldsymbol{Z}$ 是由矩阵 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 中的点确定的矩阵.如果插值点矩阵 $(\boldsymbol{X}_i,\boldsymbol{Y}_i)$ 在 $(\boldsymbol{X},\boldsymbol{Y})$ 的范围以外的值，则返回 NaN.插值点矩阵 $(\boldsymbol{X}_i,\boldsymbol{Y}_i)$ 中的 $\boldsymbol{X}_i$ 和 $\boldsymbol{Y}_i$ 可以是矩阵，也可以是向量.
ZI = interp2(Z,XI,YI)	如果节点矩阵 $(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z})$ 中的 $\boldsymbol{X}$ 的元素是 1 到 n 的自然数和 $\boldsymbol{Y}$ 的元素是 1 到 m 的自然数，即 $\boldsymbol{X}=1:n$ 和 $\boldsymbol{Y}=1:m$，这里矩阵 $\boldsymbol{Z}$ 的行数为 m，列数为 n，即[m,n] = size(Z).则用 ZI = interp2(Z,XI,YI)计算二元函数 Z 在插值点矩阵 $(\boldsymbol{X}_i,\boldsymbol{Y}_i)$ 的点 $(\boldsymbol{X}_i(k),\boldsymbol{Y}_i(k))$ 处的二元线性插值所对应的向量 $\boldsymbol{Z}_i$.
ZI = interp2 (Z,NTIMES)	ZI = interp2 (Z,NTIMES)的功能是为了 NTIMES 递归地工作，通过在每个元素之间交叉插入.interp2(Z)与 interp2(Z,1)的相同.
ZI = interp2(...,'method')	ZI = interp2(...,'method')的主要功能是用户指定二元内插值的方法.用户不输入具体的方法时，按双线性插值计算.'method'可用以下方法替换： 'nearest' 表示二元最近邻插值； 'linear' 表示双线性插值； 'cubic' 表示双三次(立方)插值； 'spline' 表示二元样条插值

说明:(1) 在使用表 6－17 中的所有的插值方法之前,必须首先将数据点(X,Y)用网格化命令[X,Y] = meshgrid(x,y)将 $\boldsymbol{x},\boldsymbol{y}$ 化为二元网格坐标 $\boldsymbol{X},\boldsymbol{Y}$,然后调用 interp2 程序.

(2) 表 6－17 中的所有的插值方法要求数据点$(\boldsymbol{X},\boldsymbol{Y})$的元素构成的序列$\{x_i\}_{i=0}^{n}$和$\{y_i\}_{i=0}^{n}$必须是单调序列,可以不是等距序列.

(3) 如果数据点是非单调的,即是随机数据点,则可以用下章的在随机数据点上的二元拟合的 MATLAB 命令 griddata.

(4) 二元最近邻插值可用于医学图像处理,双三次插值常用于图像处理(医学问题除外).

例 6.8.4 设节点(x,y,z)中的 $x = -3.0000, -1.5000, 0, 1.5000, 3.0000$, $y=x$,函数 $z=3(1-x)^2e^{-x^2-(1+y)^2}-10\left(\frac{x}{5}-x^3-y^5\right)e^{-x^2-y^2}-\frac{1}{3}e^{-(1+x)^2-y^2}$,作在节点$(x,y,z)$处 $\boldsymbol{X}=(2,3,1,7)$,$\boldsymbol{Y}=(5,2,-1,5)$的双线性插值及其图形.

解 输入程序

```
>> [x,y] = meshgrid( -3:1.5:3),
z =3 * (1 -x).^2.* exp( -(x.^2) -(y +1).^2) - 10 * (x/5 - x.^3 -
y.^5).* exp( -x.^2 -y.^2) -1/3 * exp( -(x +1).^2 -y.^2)
[xi,yi] = meshgrid([2,3,1,7],[5,2, -1,5]);
zi = interp2(x,y,z,xi,yi),
mesh(xi,yi,zi),xlabel('x'), ylabel('y'), zlabel('z')
title('z =3(1 -x)^2exp( -x^2 -(y +1)^2) -10(x/5 -x^3 -y^5)exp( -x^
2 -y^2) -1/3 exp( -(x +1)^2 -y^2) 的双线性插值图形')
```

运行后屏幕显示双线性插值(略)及其图形(见图 6－29).

例 6.8.5 设节点(x,y,z)中的 $x = -3:0.5:3$, $y=x$ 和函数 $Z=7-3x^3e^{-x^2-y^2}$值,作 Z 在插值点 $X=-3.9:0.5:5$, $Y=-4.9:0.5:4.5$ 处的二元样条插值、双三次插值和数据点的图形.

解 (1) **计算二元样条插值**. 输入程序

```
>> [x,y] =meshgrid( -3:0.5:3); z =7 -3 * x.^3 .* exp( -x.^2 - y.^2);
xi = -3.9:0.5:5;yi = -4.9:0.5:4.5; [xi,yi] = meshgrid(xi,yi);
zi = interp2(x,y,z,xi,yi,'spline'), mesh(xi,yi,zi),
hold on,
plot3(x,y,z,'r.','markersize',3 * 5), hold off
xlabel('x'), ylabel('y'),
title('z -7 -3 x^3 exp( -x^2 -y^2)的二元样条插值和数据点的图形')
```

运行后屏幕显示 Z 在插值点 $X=-3.9:0.5:5$, $Y=-4.9:0.5:4.5$ 处的二元样

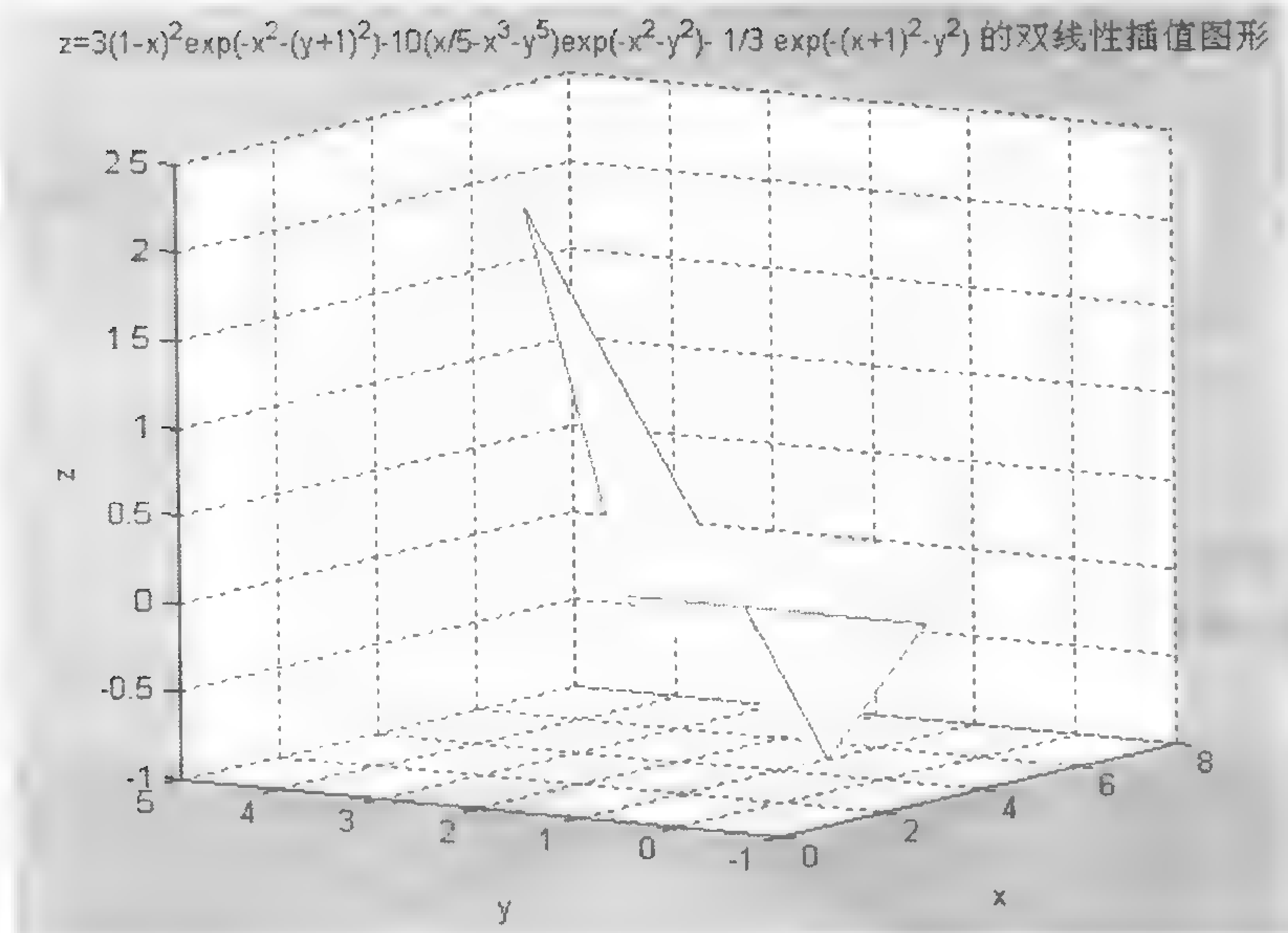


图 6－29　双线性插值的图形

条插值(略)及其图形(见图 6－30).

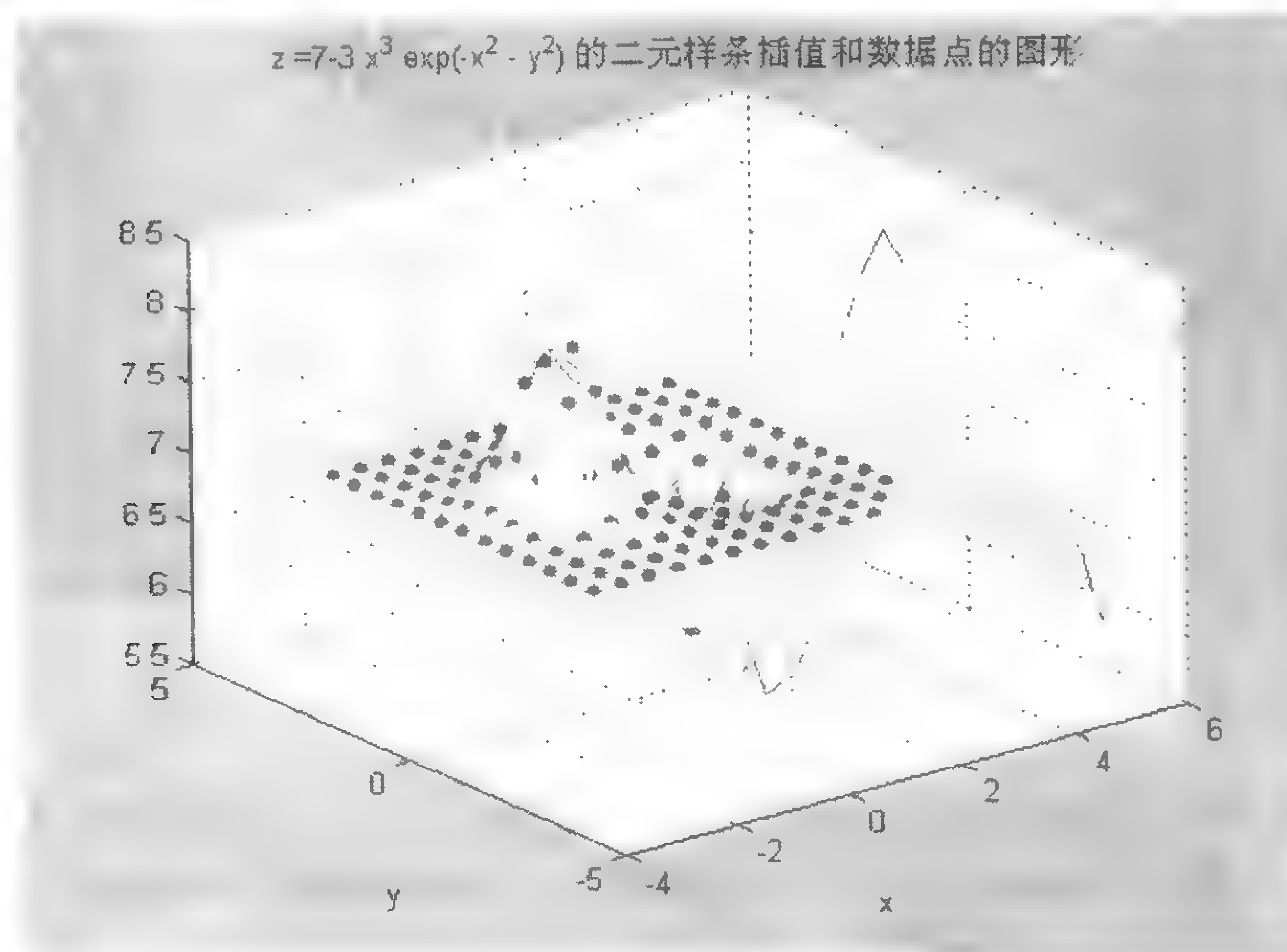


图 6－30　$Z=7-3x^3e^{-x^2-y^2}$的二元样条插值

(2) 计算双三次插值. 输入程序

```
>> [x,y] = meshgrid( -3:0.5:3);
z = 7 - 3 * x.^3 .* exp( -x.^2 - y.^2);
xi = -3.9:0.5:5; yi = -4.9:0.5:4.5; [xi,yi] = meshgrid(xi,yi);
zi = interp2(x,y,z,xi,yi,'cubic'), mesh(xi,yi,zi),hold on
plot3(x,y,z,'r.','markersize',3*5), hold off
xlabel('x'), ylabel('y'), zlabel('z'),
title('z=7-3x^3exp(-x^2-y^2) 的双三次插值和数据点(x,y,z)的图
形')
```

运行后屏幕显示 Z 在插值点 $X=-3.9:0.5:5, Y=-4.9:0.5:4.5$ 处的双三次插值(略)和数据点图形(见图 6-31)(三种方法比较留给读者)

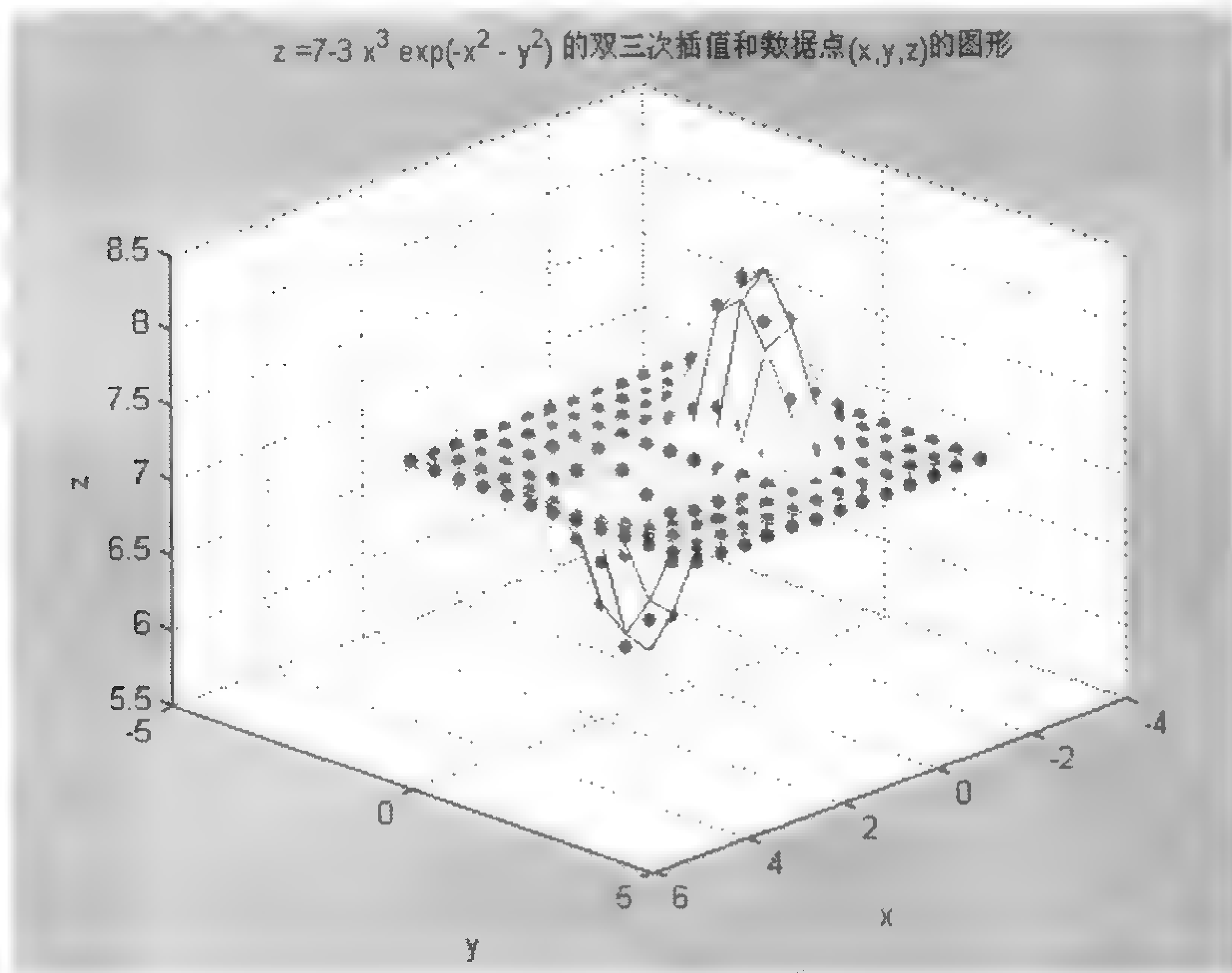


图 6-31 双三次插值和数据点图形

例 6.8.6 设节点 (x,y,z) 中的 $x=-5:0.5:5, y=x$ 和函数 $Z=7-3x^4e^{-x^2-y^2}$,作 Z 在插值点 $X=-3.9:0.5:5, Y=-4.9:0.5:4.5$ 处的双三次插值和二元最近邻插值及其图形.

解 (1) **双三次插值**. 输入程序

```
>> [x,y] = meshgrid( -5:0.5:5);
```

```
z=7-3* x.^4 .* exp(-x.^2 - y.^2);
xi = -3.9:0.5:5; yi = -4.9:0.5:4.5;
[xi,yi] =meshgrid(xi,yi);
zi = interp2(x,y,z,xi,yi,'cubic'),
mesh(xi,yi,zi)
hold on
plot3(x,y,z,'r.','markersize',3*5)
hold off
xlabel('x'), ylabel('y'), zlabel('z'),
title('z=7-3 x^4 exp(-x^2 - y^2) 的双三次插值和数据点的图形')
```

运行后屏幕显示 Z 在插值点 $X=-3.9:0.5:5, Y=-4.9:0.5:4.5$ 处的双三次插值(略)及其图形(见图 6-32).

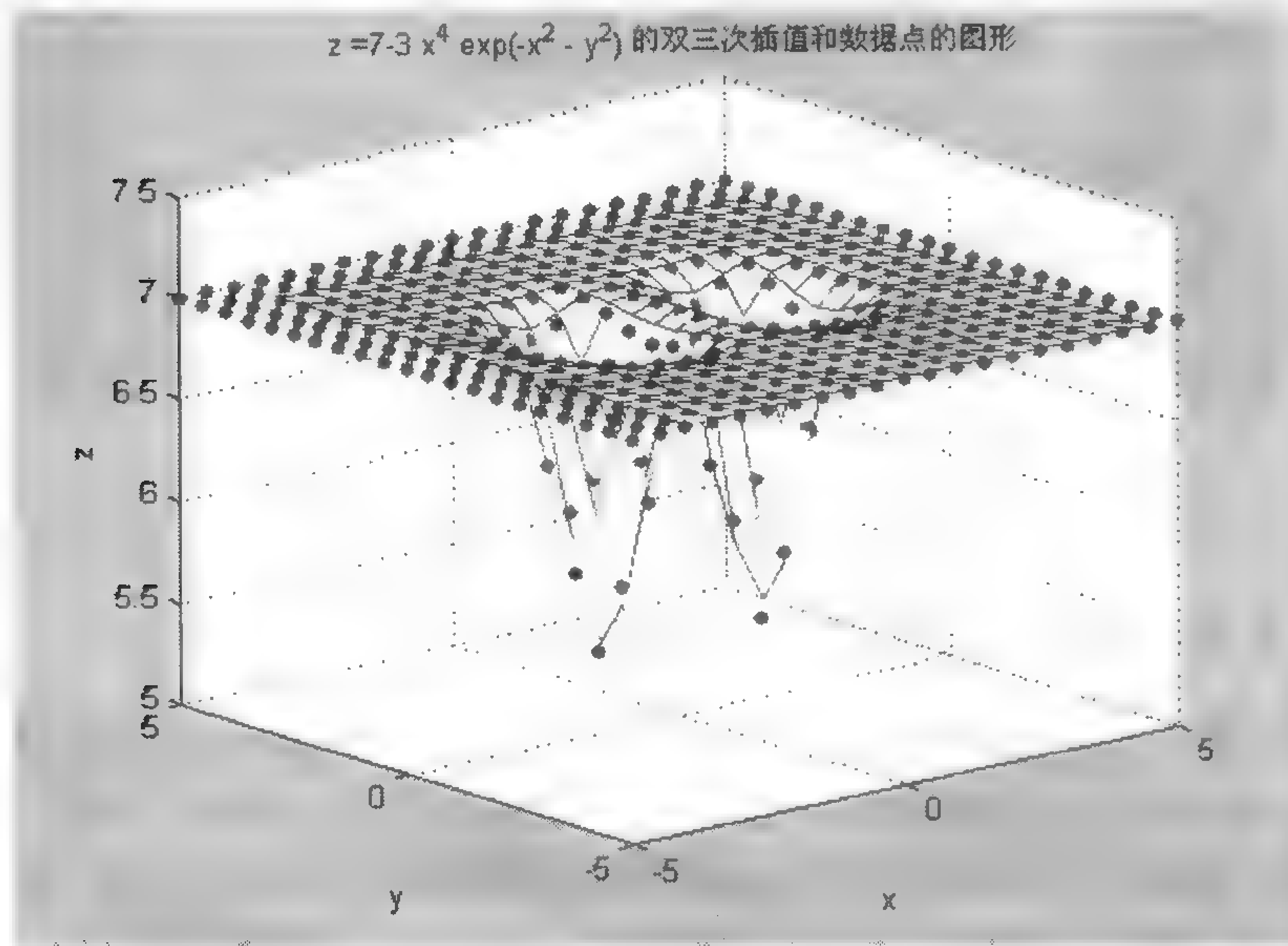


图 6-32 双三次插值和数据点的图形

(2) **二元最近邻插值.** 输入程序

```
>> [x,y] =meshgrid(-5:0.5:5); z =7-3* x.^4 .* exp(-x.^2 - y.^2);
xi = -3.9:0.5:5; yi = -4.9:0.5:4.5; [xi,yi] =meshgrid(xi,yi);
zi = interp2(x,y,z,xi,yi,'nearest'), mesh(xi,yi,zi)
hold on,plot3(x,y,z,'r.','markersize',3*5), hold off
xlabel('x'), ylabel('y'), zlabel('z')
```

```
title('z=7-3 x^3 exp(-x^2-y^2) 的二元最近邻插值和数据点的图形')
```

运行后屏幕显示 Z 在插值点 $X=-3.9:0.5:5, Y=-4.9:0.5:4.5$ 处的二元最近邻插值(略)及其图形(见图 6-33).

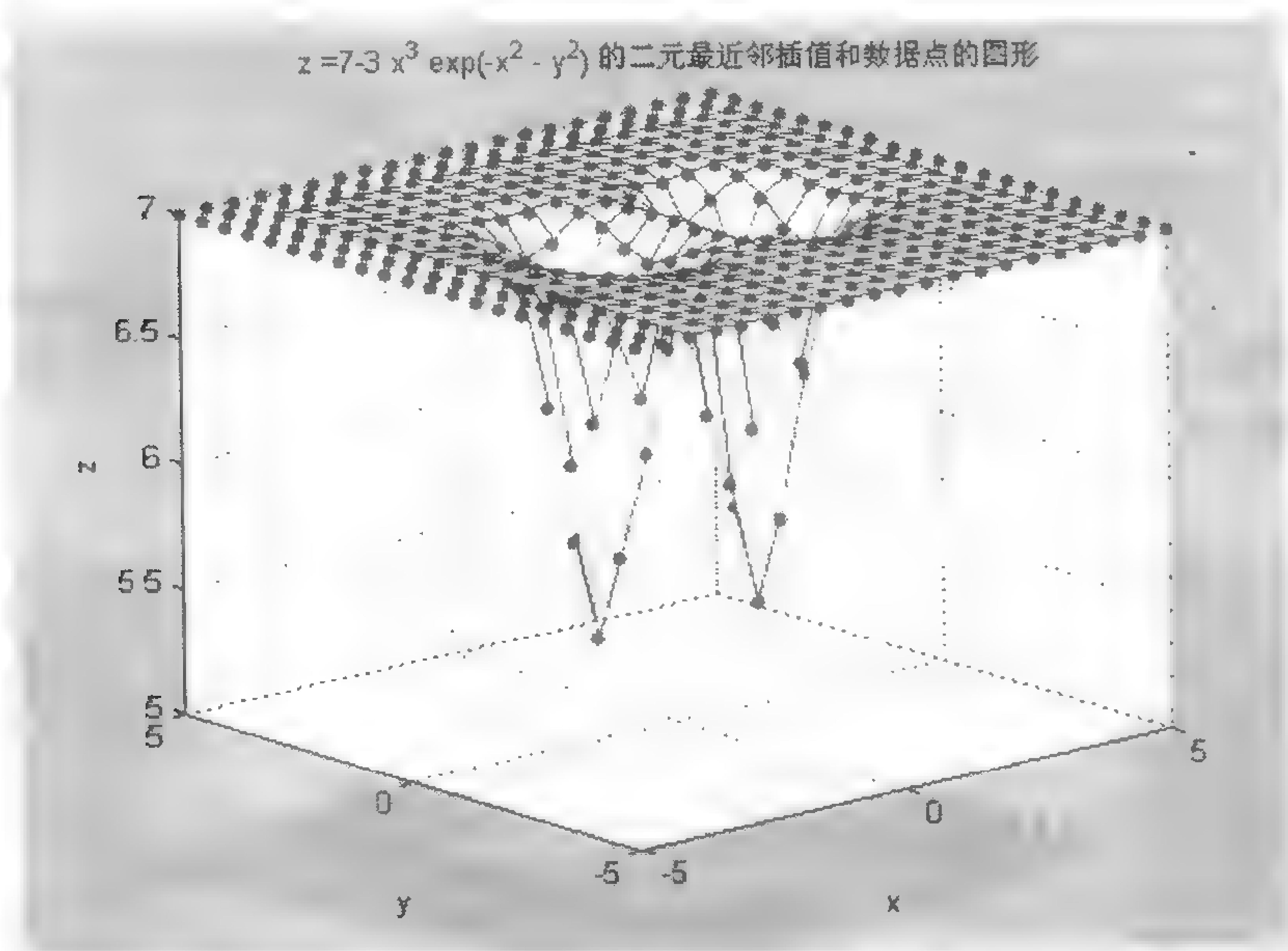


图 6-33　二元最近邻插值和数据的图形

6.8.3 三元插值及其 MATLAB 程序

常用的三元插值方法有最近邻插值、线性插值、三次插值和样条插值. 这些插值方法在 MATLAB 函数库中有编好的名为 interp3.m 的计算程序,该程序有几种调用格式,计算时根据需求选用对应的调用格式直接调用即可.这些所有的插值方法都要求利用[X,Y,Z]=MESHGRID(x,y,z)命令将节点(x,y,z)的坐标构成的向量 $\boldsymbol{x}, \boldsymbol{y}$ 和 z 转化为矩阵 $\boldsymbol{X}, \boldsymbol{Y}$ 和 $\boldsymbol{Z}$,然后用 interp3 命令计算三元插值.其中 $\boldsymbol{x}, \boldsymbol{y}$ 和 z 可以是不等距分布,但它们的元素必须是单调排列. interp3程序有几种调用格式,如表 6-18 所示.

例 6.8.7 设节点(x,y,z)的坐标为 $x=-4,0,1,12, y=-1,0,3,15, z=y$,计算函数 $V=2+xe^{-x^2-y^2-z^2}$ 在插值点 $x_i=-3:0.25:10, y_i=-3:0.25:3, z_i=-3:0.25:13$处的三元线性插值,并作其图形.

表 6-18 计算各种三元插值的 MATLAB 命令

三元插值的 MATLAB 命令 interp3	功　　能
VI = interp3(X,Y,Z,V,XI,YI,ZI)	VI = interp3(X,Y,Z,V,XI,YI,ZI)命令的主要功能是计算三元函数 V 在插值点($\boldsymbol{X}_i,\boldsymbol{Y}_i,\boldsymbol{Z}_i$)的元素($\boldsymbol{X}_i(k),\boldsymbol{Y}_i(k),\boldsymbol{Z}_i(k)$)处的三元线性插值所对应的向量 $\boldsymbol{V}_i$，其中 $\boldsymbol{X}_i,\boldsymbol{Y}_i,\boldsymbol{Z}_i$ 必须是同型矩阵或向量，但是可以不相等.($\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}$)是由节点($\boldsymbol{X}_1,\boldsymbol{Y}_2,\boldsymbol{Z}_3$)通过命令 meshgrid 转换的三元网格坐标.$\boldsymbol{V}$ 的值由($\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}$)确定.如果输入的插值点($\boldsymbol{X}_i,\boldsymbol{Y}_i,\boldsymbol{Z}_i$)的元素($\boldsymbol{X}_i(k),\boldsymbol{Y}_i(k),\boldsymbol{Z}_i(k)$)在($\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}$)所确定的范围以外，则返回的值是 NaN.
VI = interp3(V,XI,YI,ZI)	如果节点矩阵($\boldsymbol{X}_1,\boldsymbol{Y}_2,\boldsymbol{Z}_3$)中的 $\boldsymbol{X}_1$ 的元素是 1 到 N 的自然数，$\boldsymbol{Y}_2$ 的元素是 1 到 M 的自然数，$\boldsymbol{Z}_3$ 的元素是 1 到 P 的自然数即 $\boldsymbol{X}_1=1:N$，$\boldsymbol{Y}_2=1:M$，$\boldsymbol{Z}_3=1:P$ 这里[M,N,P] = SIZE(V).则用此命令计算三元函数 Z 在插值点矩阵($\boldsymbol{X}_i,\boldsymbol{Y}_i,\boldsymbol{Z}_i$)处的三元线性插值所对应的向量 $\boldsymbol{V}_i$.
VI = interp3(V,NTIMES)	VI = interp3(V,NTIMES)的功能是为了 NTIMES 递归地工作，通过在每个元素之间交叉插人. interp3(V)与 interp3(V,1)相同.
VI = interp3(...,'method')	VI = interp3(...,'method')的主要功能是用户指定三元内插值的方法.用户不输入具体的方法时，按线性插值计算.'method'可用以下方法替换： 'nearest' 表示最近邻插值； 'linear' 表示双线性插值； 'cubic' 表示三次插值； 'spline' 表示样条插值.

解 输入程序

```
>> x=[-4,0,1,12];y=[-1,0,3,15];z=y; [X,Y,Z]=meshgrid(x,y,z);
V=2+X .* exp(-X.^2 - Y.^2 - Z.^2);
[xi,yi,zi]=meshgrid(-3:.25:10,-3:.25:3,-3:.25:13);
vi=interp3(X,Y,Z,V,xi,yi,zi), slice(xi,yi,zi,vi,[-1 6 9.5],9,
[-2 .2 9]),
shading flat,lighting flat
xlabel('x'), ylabel('y'), zlabel('z'),
title('V=2+x exp(-x^2 - y^2 -z^2) 的三元线性插值图形')
hold on,colorbar('horiz'), view([-30  45])
```

运行后屏幕显示三元线性插值(略)及其图形(如图 6-34 所示).

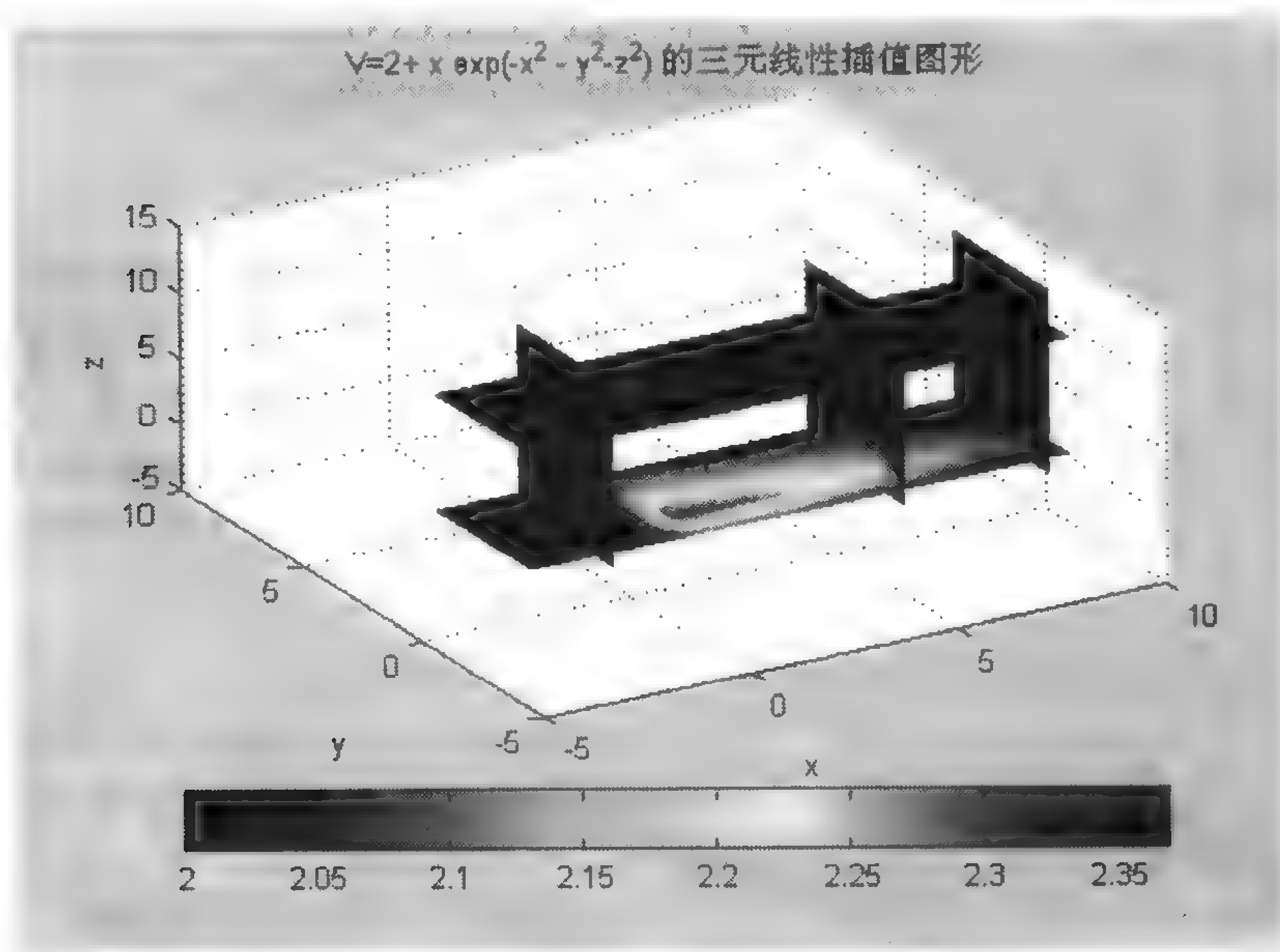


图 6-34 $V=2+xe^{-x^2-y^2-z^2}$ 的三元线性插值图形

例 6.8.8 取 $n=10$,作函数 flow 在插值点 $x_i=0.1:0.25:10$,$y_i=-3:0.25:3$,$z_i=y_i$ 处的三元三次样条插值及其图形.

解 输入程序

```
>> [x,y,z,v]=flow(10);
[xi,yi,zi]=meshgrid(.1:.25:10,-3:.25:3,-3:.25:3);
vi=interp3(x,y,z,v,xi,yi,zi,'spline'); % vi is 25-by-40-by-25
slice(xi,yi,zi,vi,[2.5 7.5],0.1,[-1.2 1.5]),
```

```
shading flat,xlabel('x'), ylabel('y'), zlabel('z'),
title('flow 的三元三次样条插值图形')
hold on,colorbar('horiz')
```

运行后屏幕显示三元三次样条插值(略)及其图形(如图 6-35 所示).

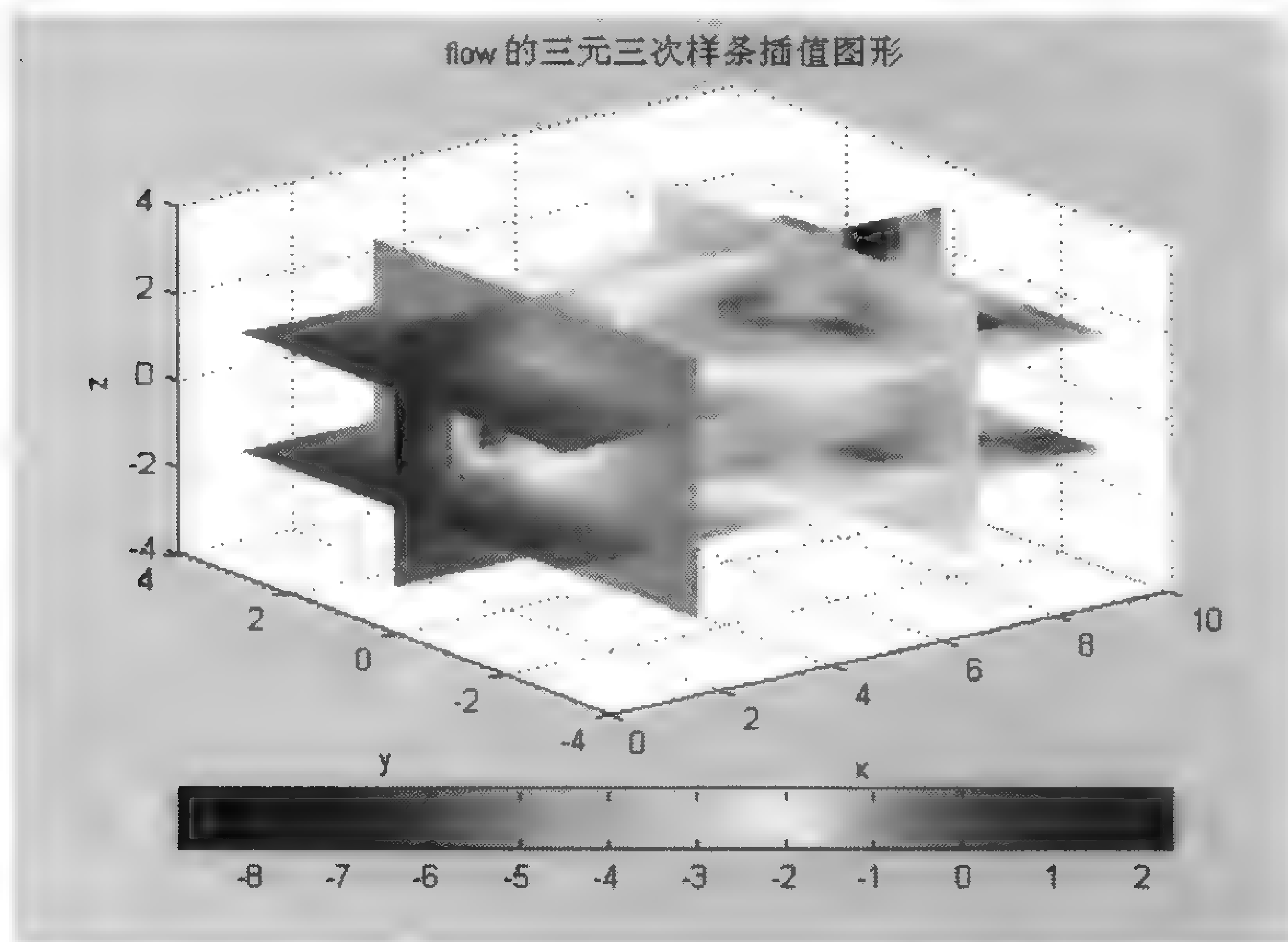
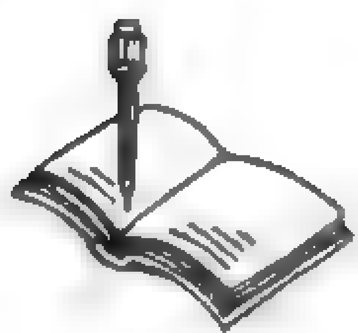


图 6-35 函数 flow 的三元三次样条插值及其图形



习 题 6.8

1. 设节点(x,y,z)中的 $x=-3.000\,0,-1.500\,0,0,1.500\,0,3.000\,0$, $y=x$, 函数 $z=2(1-x)^2e^{-x^2-(1+y)}-8\left(\frac{x}{5}-x^3-y\right)e^{-x^2-y^2}-\frac{1}{4}e^{-(1+x)^2-y}$, 计算在节点$(x,y,z)$处 $X=(2,3,1,7)$, $Y=(5,2,-1,5)$的双线性插值、三次插值、样条插值和最近邻插值及其图形.

2. 设节点(x,y,z)中的 $x=-5:0.2:5$ $y=x$, 函数 $z=5-4x^4e^{-x^2-y^2}$, 计算在 $X=-4.5:0.3:4.5$, $Y=X$ 处的表示二元样条插值的双线性插值、三次插值和最近邻插值及其图形.

3. 计算函数 $U=3+2x^3e^{-x^2-y^2-z^2}$在 $x=0,1,2,-4$, $y=0,-1,3,5$, $z=y$ 处的函数值,并作图.

4. 设节点(x,y,z)的坐标为 $x=-4,0,1,12$, $y=-1,0,3,15$, $z=y$, 计算函数 $V=5+2xyz\,e^{-x^2-y^2-z^2}$在插值点 $x_i=0.1:0.5:10$, $y_i=-0.5:0.25:13$, $z_i=y_i$ 处的三元线性插值、三次插值、

样条插值和最近邻插值，并作其图形.

5. 取 $n=9,5$，作函数 flow 在插值点 $x_i=0.1:0.25:10$，$y_i=-3:0.25:3$，$z_i=y_i$ 处的三元线性插值、三次插值、样条插值和最近邻插值及其图形.

第七章 函数逼近与曲线(面)拟合

在众多科学技术、科学研究过程中,由一组实验数据$\{(x_i,y_i),i=1,2,\cdots,n\}$选择一个较简单的函数$f(x)$(如多项式),在一定准则下最接近这组数据,这就是曲线拟合的问题. 与曲线拟合相近的问题是函数逼近,所谓函数逼近就是:已知一个较为复杂的连续函数 $y(x),x\in[a,b]$,要求选择一个较简单的函数$f(x)$,在一定准则下最接近 $y(x)$. 本章主要介绍曲线(面、体)的拟合和函数的最佳逼近及其用 MATLAB 软件进行计算等问题.

7.1 曲线拟合、误差及其 MATLAB 程序

曲线拟合(即数据拟合)在实际中有着广泛的应用. 曲线拟合问题是这样提出的,我们通过测量或观察等方法获得一组看上去杂乱无章的实验数据(x_i,y_i),即平面上的 n 个点$(x_i,y_i),i=1,2,\cdots,n,x_i$互不相同(如图 7-1 所示),我们希望从中找出某种规律,即寻求一个函数(曲线)$y=f(x)$,使$f(x)$在某种准则下与所有数据点(x_i,y_i)最为接近,即曲线拟合得最好.

虽然第六章介绍的插值方法是处理数据近似的一种数值方法,但是用来解决这里提出的问题具有明显的缺陷. 首先,这类问题的数据(x_i,y_i)是通过观察或测量等手段得到的实验数据,其本身往往带有测量误差,个别数据的误差可能还很大,如果要求所求的近似函数的曲线精确无误地通过每一个数据点(x_i,y_i),就会使曲线保留所有的测量误差,从而失去原数据表示的规律. 其次,实验数据往往很多,用插值方法得到的近似函数明显地缺乏实用价值.

曲线拟合是解决此类问题的一种常用方法. 在曲线拟合时,不要求曲线$f(x)$严格地经过所有的数据点(x_i,y_i),只要求拟合函数$f(x)$在 x_i 处的误差(又称偏差或残差)

$$\delta_i=f(x_i)-y_i,\ i=1,2,\cdots,n \tag{7.1}$$

不全为零. 但是,为了使近似曲线$f(x)$能够尽可能地反映所给点的变化趋势,要求误差适当地小是必要的. 达到这一目标的方法很多,例如,通过求下列四种误差

(1) 最大误差

$$E_\infty=\max_{1\le i\le n}|\delta_i|=\max_{1\le i\le n}|f(x_i)-y_i|, \tag{7.2}$$

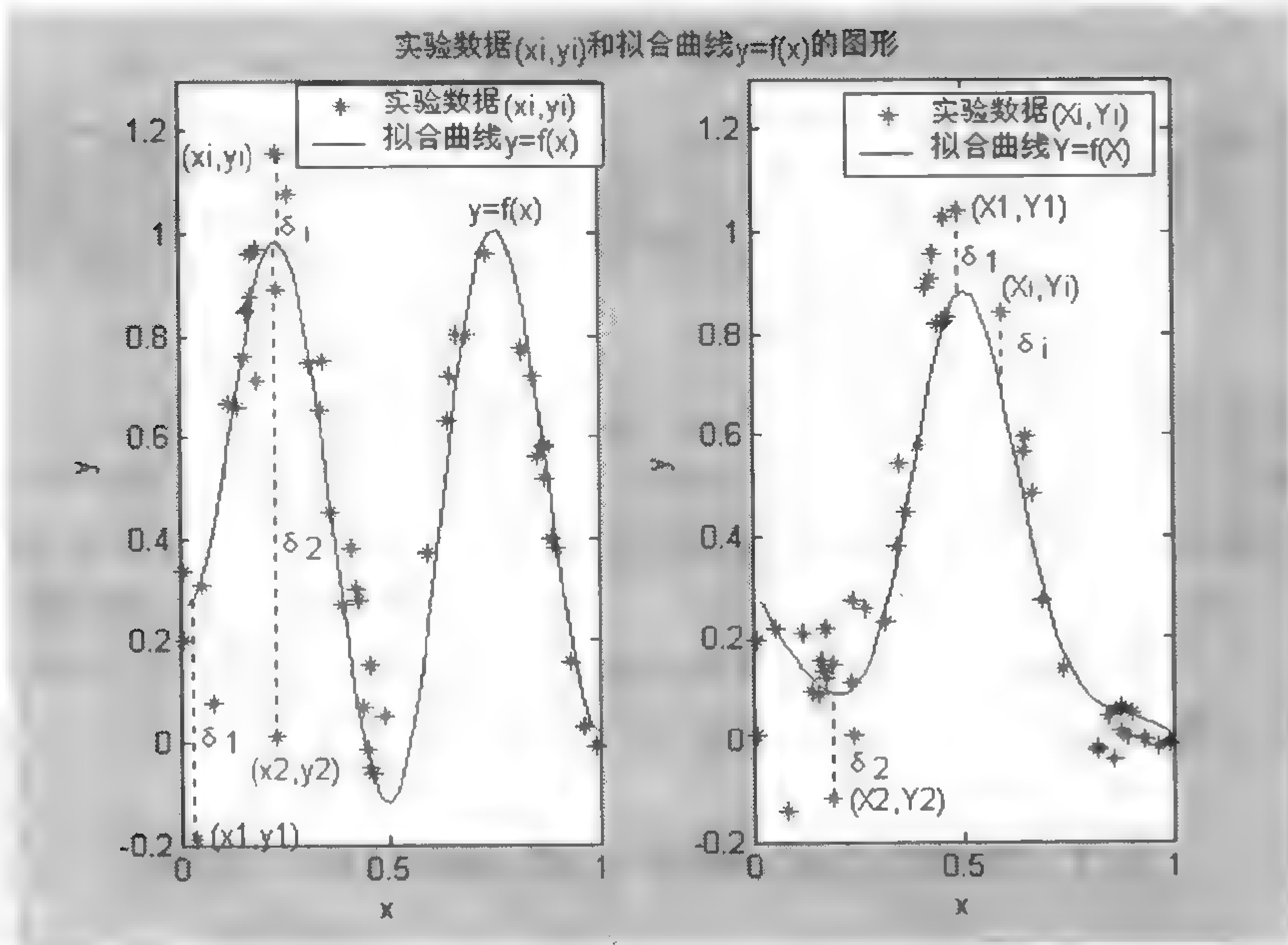


图 7-1 实验数据和曲线拟合示意图

(2) 平均误差

$$E_1=\frac{1}{n}\sum_{i=1}^{n}|\delta_i|=\frac{1}{n}\sum_{i=1}^{n}|f(x_i)-y_i|, \tag{7.3}$$

(3) 均方根误差

$$E_2=\left(\frac{1}{n}\sum_{i=1}^{n}|\delta_i|^2\right)^{\frac{1}{2}}=\left(\frac{1}{n}\sum_{i=1}^{n}|f(x_i)-y_i|^2\right)^{\frac{1}{2}} \tag{7.4}$$

(4) 误差平方和

$$E=\sum_{i=1}^{n}\delta_i^2=\sum_{i=1}^{n}[f(x_i)-y_i]^2 \tag{7.5}$$

之一的最小值等,用这四种方法就可以分别得到在这四种准则下的四条最佳拟合曲线. 使其误差平方和最小的方法称为**最小二乘准则**. 由于最小二乘准则容易进行最小化计算,所以通常采用它.

如果给定拟合曲线的函数 $f(x)$ 的系数,我们还可以用上面的最大误差、平均误差和均方根误差等方法估计 $f(x)$ 与实验数据的误差. 下面的例题 7.1.1 展示了当给定一个函数和一组数据以后,如何使用这些误差公式和 MATLAB 软件计算误差.

例 7.1.1 已知函数 $y=f(x)=5x^3-14x+7\sin^2(2\pi x)$ 和一组数据 (x_i, y_i) 列入表 7-1 中,比较最大误差、平均误差、均方根误差和误差平方和.

表 7-1 例 7.1.1 的一组数据 (x_i, y_i)

x_i	-2.5	-1.7	-1.1	-0.8	0	0.1	0.5	3.6
y_i	-43.50	5.69	11.34	14.16	0	1.02	-6.37	185.84

解 由给定的函数和数据，根据(7.1)至(7.5)式编写下列程序，并在 MATLAB 工作窗口输入

```
>> x=[-2.5,-1.7,-1.1,-0.8,0,0.1,0.5,3.6]; n=length(x);
y=[-43.50,5.69,11.34,14.16,0,1.02,-6.37,185.84];
f=5.*x.^3-14.*x+7.*(sin(2*pi*x)).^2; fy=abs(f-y);
fy2=fy.^2; [x',y',f',fy',fy2'], Ew=max(fy),
E1=sum(fy)/n, E2=sqrt((sum(fy2))/n), E=sum(fy2)
```

运行后屏幕显示数据 (x_i, y_i) 的横坐标向量 $\boldsymbol{x}$，纵坐标向量 $\boldsymbol{y}$，函数值 f，绝对误差 fy，绝对误差平方 fy_2，最大误差 E_w，平均误差 E_1，均方根误差 E_2 和误差平方和 E 如下

```
      x          y          f          fy         fy2
  -2.5000   -43.5000   -43.1250     0.3750     0.1406
  -1.7000     5.6900     5.5666     0.1234     0.0152
  -1.1000    11.3400    11.1634     0.1766     0.0312
  -0.8000    14.1600    14.9716     0.8116     0.6586
        0          0          0          0          0
   0.1000     1.0200     1.0234     0.0034     0.0000
   0.5000    -6.3700    -6.3750    0.0050     0.0000
   3.6000   185.8400   185.2984     0.5416     0.2933
    Ew =        E1 =        E2 =        E =

      0.8116      0.2546      0.3773    1.1390
```

由此可见，误差平方和 E 最大，其次是最大误差 E_w，我们可以根据拟合精度和最大误差 E_w 来决定是否删除奇异点. 因为平均误差 E_1 最小且计算简单，所以经常被使用估计误差. 均方根误差 E_2 通常用于需要考虑误差的统计特征的情况.



习 题 7.1

1. 已知函数 $y=f(x)=4x+7\sin^2(2\pi x)$ 和一组数据 (x_i, y_i) 列入下表中，比较最大误差、平均误差、均方根误差和误差平方和.

x_i	-2.5	-1.7	-1.1	-0.8	0	0.1	0.5	3.6
y_i	-43.50	5.69	11.34	14.16	0	1.02	-6.37	185.84

2. 已知函数 $y=5.0911x^3-14.1905x^2+6.4102x-8.2574$ 和一组数据(x_i,y_i)列入下表中,比较最大误差、平均误差、均方根误差和误差平方和.

x_i	-2.5	-1.7	-1.1	-0.8	0	0.1	1.5	2.7	3.6
y_i	-192.9	-85.50	-36.15	-26.52	-9.10	-8.43	-13.12	6.50	68.04

7.2 曲线拟合的线性最小二乘法及其 MATLAB 程序

用最小二乘准则选择近似函数的方法称为**最小二乘法**,其中线性最小二乘法是解决曲线拟合最常用的方法,基本思路是,令

$$f(x)=a_1r_1(x)+a_2r_2(x)+\cdots+a_mr_m(x), \tag{7.6}$$

其中 $r_k(x)$ 是事先选定的一组函数,$a_k(k=1,2,\cdots,m,m<n)$ 是待定系数. 拟合准则是**最小二乘准则**,其几何意义是使 n 个点 $(x_i,y_i),i=1,2,\cdots,n$ 与对应点 $(x_i,f(x_i))$ 的距离 δ_i 的平方和(7.5)式最小(参考图 7-1). 由于(7.5)式是系数 a_k 的函数,记

$$J(a_1,\cdots,a_m)=\sum_{i=1}^{n}\delta_i^2=\sum_{i=1}^{n}[f(x_i)-y_i]^2. \tag{7.7}$$

为求 $a_1,a_2,\cdots,a_m$ 使 J 达到最小,只需利用极值的必要条件 $\frac{\partial J}{\partial a_k}=0\ (k=1,2,\cdots,m)$,得到关于 $a_1,a_2,\cdots,a_m$ 的线性方程组

$$\begin{cases}\sum_{i=1}^{n} r_1(x_i)\left[\sum_{k=1}^{m} a_kr_k(x_i)-y_i\right]=0,\\ \cdots\cdots\cdots\cdots \\ \sum_{i=1}^{n} r_m(x_i)\left[\sum_{k=1}^{m} a_kr_k(x_i)-y_i\right]=0.\end{cases} \tag{7.8}$$

记

$$\boldsymbol{R}=\begin{pmatrix} r_1(x_1) & \cdots & r_m(x_1)\\ \vdots & & \vdots \\ r_1(x_n) & \cdots & r_m(x_n)\end{pmatrix}_{n\times m},\boldsymbol{A}=(a_1,\cdots,a_m)^{\mathrm{T}},\ \boldsymbol{y}=(y_1,\cdots,y_n)^{\mathrm{T}}.$$

方程组(7.8)可表示为

$$\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{A}=\boldsymbol{R}^{\mathrm{T}}\boldsymbol{y}. \tag{7.9}$$

当$\{r_1(x),\cdots,r_m(x)\}$线性无关时,$\boldsymbol{R}$ 是满秩矩阵,故 $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}$ 可逆,于是方程组(7.9)有唯一解

$$\boldsymbol{A}=(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R})^{-1}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{y}. \tag{7.10}$$

可以看出,只要 $f(x)$ 关于待定系数 $a_1,a_2,\cdots,a_m$ 是线性的,在最小二乘准则(7.5)下得到的方程组(7.8)关于 $a_1,a_2,\cdots,a_m$ 也一定是线性的,故称**线性最小二乘法**.

例 7.2.1 给出一组数据点(x_i,y_i)列入表 7-2 中,试用线性最小二乘法求拟合曲线,并用(7.2),(7.3)和(7.4)式估计其误差,作出拟合曲线.

表 7-2 例 7.2.1 的一组数据(x_i,y_i)

x_i	-2.5	-1.7	-1.1	-0.8	0	0.1	1.5	2.7	3.6
y_i	-192.9	-85.50	-36.15	-26.52	-9.10	-8.43	-13.12	6.50	68.04

解 (1) 首先根据表 7-2 给出的数据点(x_i,y_i),用下列 MATLAB 程序画出散点图.

在 MATLAB 工作窗口输入程序

```
>> x=[-2.5 -1.7 -1.1 -0.8 0 0.1 1.5 2.7 3.6];
y=[-192.9 -85.50 -36.15 -26.52 -9.10 -8.43 -13.12 6.50
68.04];
plot(x,y,'r*'),
legend('实验数据(xi,yi)')
xlabel('x'),ylabel('y'),
title('例7.2.1 的数据点(xi,yi)的散点图')
```

运行后屏幕显示数据的散点图,见图 7-2.

(2) 因为数据的散点图 7-2 的变化趋势与三次多项式很接近,所以选取一组函数 $1,x,x^2,x^3$,令

$$f(x)=a_1x^3+a_2x^2+a_3x+a_4, \tag{7.11}$$

其中 a_k 是待定系数$(k=1,2,3,4)$.

(3) 用最小二乘准则求待定系数 $a_k(k=1,2,3,4)$.

将表 7-2 给出的数据(x_i,y_i), $i=1,2,\cdots,9$ 代入(7.11)式,编写下列 MATLAB 程序计算$f(x)$在(x_i,y_i)处的函数值,即输入程序

```
>> syms a1 a2 a3 a4
x=[-2.5 -1.7 -1.1 -0.8 0 0.1 1.5 2.7 3.6];
fi=a1.*x.^3+a2.*x.^2+a3.*x+a4
```

运行后屏幕显示关于 a_1,a_2,a_3 和 a_4 的线性方程组

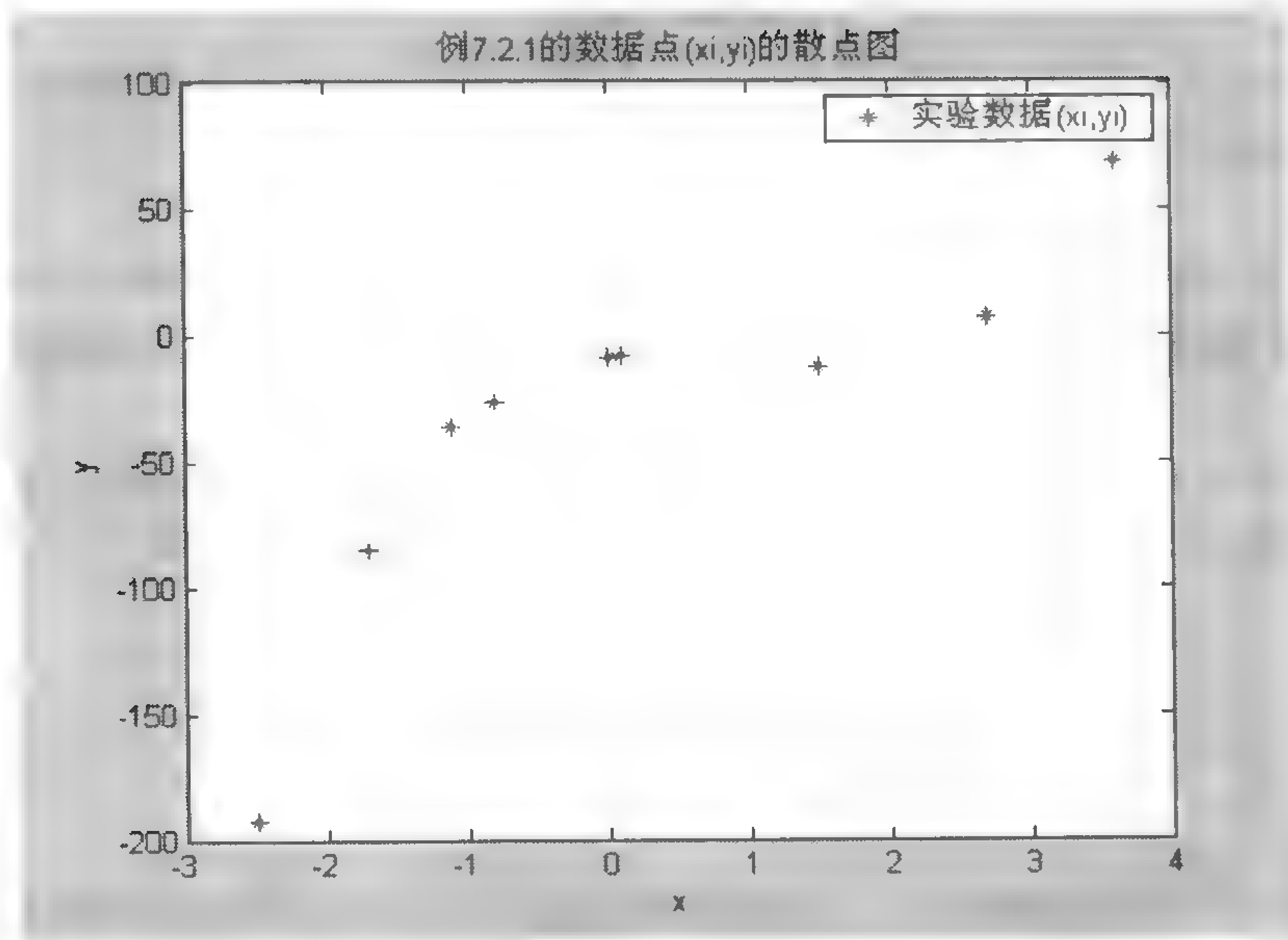


图 7-2 例 7.2.1 的数据点的散点图

```
fi =
[ -125/8*a1+25/4*a2-5/2*a3+a4, -4913/1000*a1+289/100*a2-17/10*a3+a4, -1331/1000*a1+121/100*a2-11/10*a3+a4, -64/125*a1+16/25*a2-4/5*a3+a4,a4,1/1000*a1+1/100*a2+1/10*a3+a4,27/8*a1+9/4*a2+3/2*a3+a4,19683/1000*a1+729/100*a2+27/10*a3+a4,5832/125*a1+324/25*a2+18/5*a3+a4]
```

根据(7.5)式,编写构造误差平方和 $J(a_1,\cdots,a_m)=\sum_{i=1}^{n}\delta_i^2=\sum_{i=1}^{n}[f(x_i)-y_i]^2$ 的 MATLAB 程序

```
>> y=[-192.9 -85.50 -36.15 -26.52 -9.10 -8.43 -13.12 6.50 68.04];
fi=[-125/8*a1+25/4*a2-5/2*a3+a4, -4913/1000*a1+289/100*a2-17/10*a3+a4, -1331/1000*a1+121/100*a2-11/10*a3+a4,-64/125*a1+16/25*a2-4/5*a3+a4,a4,1/1000*a1+1/100*a2+1/10*a3+a4,27/8*a1+9/4*a2+3/2*a3+a4,19683/1000*a1+729/100*a2+27/10*a3+a4,5832/125*a1+324/25*a2+18/5*a3+a4];
```

```
fy = fi - y; fy2 = fy.^2; J = sum(fy.^2)
```

运行后屏幕显示误差平方和如下

```
J =
(-125/8*a1+25/4*a2-5/2*a3+a4+1929/10)^2+(-4913/
1000*a1+289/100*a2-17/10*a3+a4+171/2)^2+(-1331/1000*a1+
121/100*a2-11/10*a3+a4+723/20)^2+(-64/125*a1+16/25*a2-4/5
*a3+a4+663/25)^2+(a4+91/10)^2+(1/1000*a1+1/100*a2+1/10*a3
+a4+843/100)^2+(27/8*a1+9/4*a2+3/2*a3+a4+328/25)^2+(19683/
1000*a1+729/100*a2+27/10*a3+a4-13/2)^2+(5832/125*a1+324/25
*a2+18/5*a3+a4-1701/25)^2
```

为求 a_1,a_2,a_3,a_4 使 J 达到最小,只需利用极值的必要条件 $\frac{\partial J}{\partial a_k}=0$ $(k=1,2,3,4)$,得到关于 a_1,a_2,a_3,a_4 的线性方程组,这可以由下面的 MATLAB 程序完成,即输入程序

```
>> syms a1 a2 a3 a4
J=(-125/8*a1+25/4*a2-5/2*a3+a4+1929/10)^2+(-4913/
1000*a1+289/100*a2-17/10*a3+a4...+171/2)^2+(-1331/1000*a1+
121/100*a2-11/10*a3+a4+723/20)^2+(-64/125*a1+16/25*a2-4/5
*a3+a4+663/25)^2+(a4+91/10)^2+(1/1000*a1+1/100*a2+1/10*a3
+a4+843/100)^2+(27/8*a1+9/4*a2+3/2*a3+a4+328/25)^2+(19683/
1000*a1+729/100*a2+27/10*a3+a4-13/2)^2+(5832/125*a1+324/25
*a2+18/5*a3+a4-1701/25)^2;
Ja1=diff(J,a1);Ja2=diff(J,a2);Ja3=diff(J,a3);Ja4=diff(J,
a4);
Ja11=simple(Ja1),Ja21=simple(Ja2),Ja31=simple(Ja3),Ja41=
simple(Ja4),
```

运行后屏幕显示 J 分别对 a_1,a_2,a_3,a_4 的偏导数如下

```
Ja11 =
    56918107/10000*a1+32097579/25000*a2+1377283/2500*
a3+23667/250*a4-8442429/625
Ja21 =
    32097579/25000*a1+1377283/2500*a2+23667/250*a3+
67*a4+767319/625
Ja31 =
    1377283/2500*a1+23667/250*a2+67*a3+18/5*a4-
232638/125
```

```
Ja41 =
     23667/250*a1+67*a2+18/5*a3+18*a4+14859/25
```

解线性方程组 $Ja_{11}=0, Ja_{21}=0, Ja_{31}=0, Ja_{41}=0$，输入下列程序

```
>> A = [56918107/10000, 32097579/25000, 1377283/2500, 23667/250; 32097579/25000, 1377283/2500, 23667/250, 67; 1377283/2500, 23667/250, 67, 18/5; 23667/250, 67, 18/5, 18];
B = [8442429/625, -767319/625, 232638/125, -14859/25];
C = B/A, f = poly2sym(C)
```

运行后屏幕显示拟合函数 f 及其系数 C 如下

```
C =    5.0911   -14.1905    6.4102    -8.2574
f = 7165036958457 59/140737488355328*x^3 - 7988544102557579/562949953421312*x^2 + 1804307491277693/281474976710656*x - 4648521160813215/562949953421312
```

故所求的拟合曲线为

$$f(x)=5.0911x^3-14.1905x^2+6.4102x-8.2574 \tag{7.12}$$

(4) 根据(7.2)、(7.3)和(7.4)式编写下面的 MATLAB 程序估计其误差，并作出拟合曲线和数据的图形. 输入程序

```
>> xi = [-2.5 -1.7  -1.1  -0.8  0  0.1  1.5  2.7  3.6];
   y = [-192.9 -85.50 -36.15 -26.52 -9.10 -8.43 -13.12  6.50  68.04];
n = length(xi);
f = 5.0911.*xi.^3 -14.1905.*xi.^2 +6.4102.*xi -8.2574;
x = -2.5:0.01:3.6;
F = 5.0911.*x.^3 -14.1905.*x.^2 +6.4102.*x -8.2574;
fy = abs(f-y); fy2 = fy.^2; Ew = max(fy),
E1 = sum(fy)/n, E2 = sqrt((sum(fy2))/n)
plot(xi,y,'r*'), hold on, plot(x,F,'b-'), hold off
legend('数据点(xi,yi)','拟合曲线 y = f(x)'),
xlabel('x'), ylabel('y'),
title('例 7.2.1 的数据点(xi,yi)和拟合曲线 y = f(x)的图形')
```

运行后屏幕显示数据 (x_i,y_i) 与拟合函数 f 的最大误差 E_w，平均误差 E_1 和均方根误差 E_2 及其数据点 (x_i,y_i) 和拟合曲线 $y=f(x)$ 的图形，见图 7-3.

```
Ew =          E1 =          E2 =
   3.105 4       0.903 4       1.240 9
```

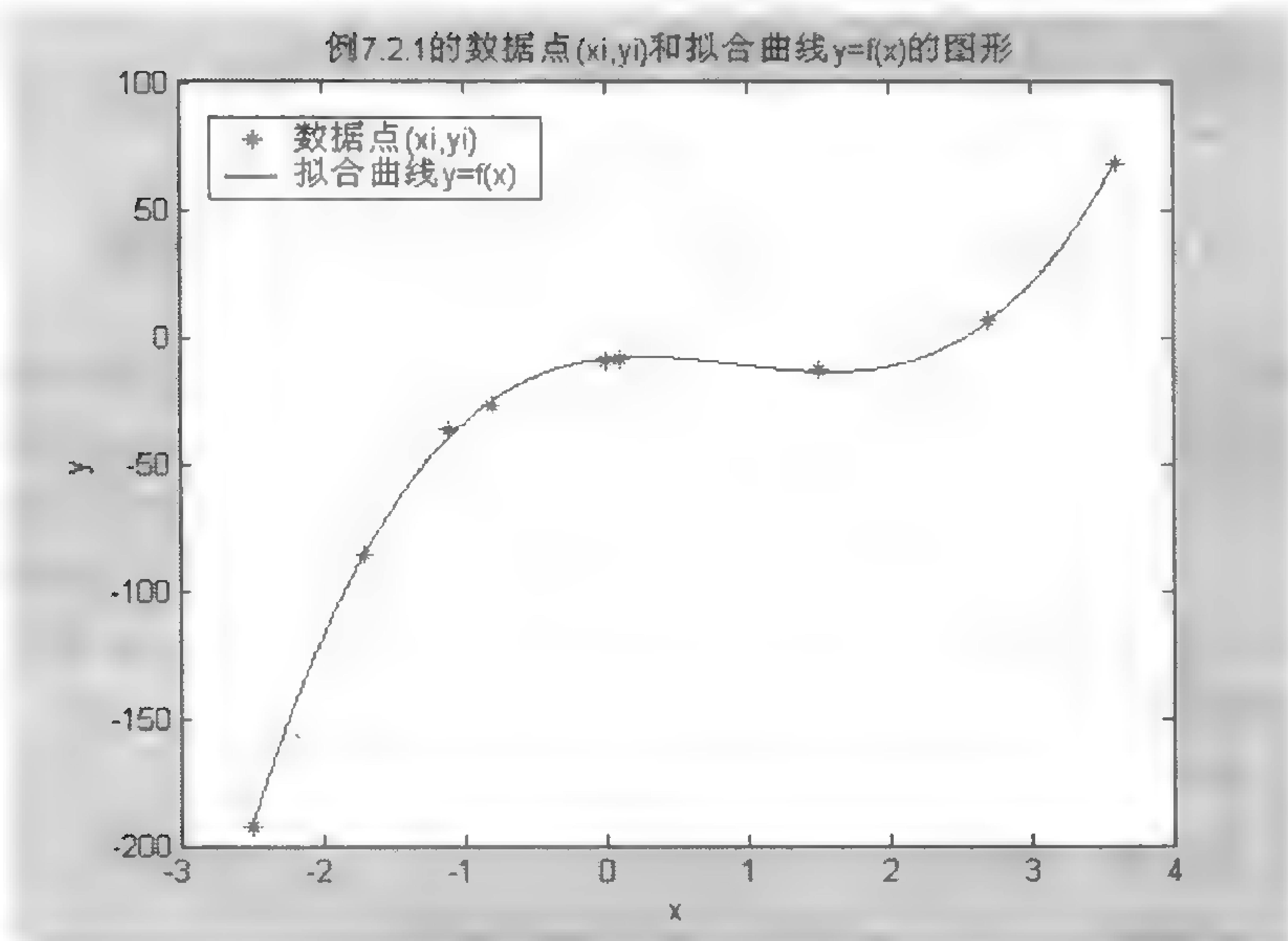
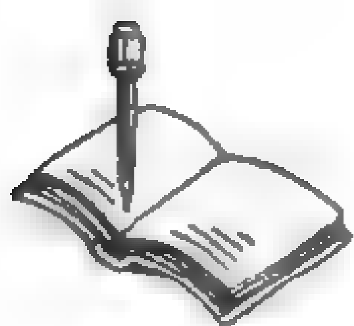


图 7-3 例 7.2.1 数据散点图和拟合曲线



习题 7.2

1. 给出一组数据点(x_i,y_i)列入下表中，试用线性最小二乘法求拟合曲线，并用(7.2)、(7.3)和(7.4)式估计其误差，作出拟合曲线.

x_i	−3.5	−2.7	−1.1	−0.8	0	0.1	1.5	2.7	3.6
y_i	−92.9	−85.56	−36.15	−26.52	−9.16	−8.43	−13.12	6.59	68.64

2. 用$y=\sqrt{x}$在$x=0,1,4,9,16$产生 5 个节点$P_1,\cdots,P_5$. 用不同的节点构造拟合曲线来计算$x=5$处的插值(如用$P_1\cdots P_5$;$P_1\cdots P_4$;$P_2\cdots P_4$等)，与精确值比较并进行分析.

3. 用给定的多项式，如$y=21x^3-6x^2+5x-8$，产生一组数据(x_i,y_i)，$i=1,2,\cdots,n$，再在y_i上添加随机干扰(可用 rand 产生(0,1)均匀分布随机数，或用 randn 产生$N(0,1)$分布随机数)，然后用x_i和添加了随机干扰y_i的数据作 3 次多项式拟合，与原系数比较. 如果作 2 或 4 次多项式拟合，结果如何？

4. 弹簧在力F的作用下伸长x，一定范围内服从胡克定律：F与x成正比，即$F=kx$，k为弹性系数. 现在得到下面一组x,F数据，请在(x,F)坐标下作图，并观察当F大到一定数值

后,是否服从这个定律. 试由数据确定 k,并给出不服从胡克定律时的近似公式.

x	1	2	4	7	9	12	13	15	17
F	1.5	3.9	6.6	11.7	15.6	18.8	19.6	20.6	21.1

7.3 函数 $r_k(x)$ 的选取及其 MATLAB 程序

面对一组数据 (x_i,y_i), $i=1,2,\cdots,n$,用线性最小二乘法作曲线拟合时,首要的、也是关键的一步是恰当地选取一组函数 $r_1(x),\cdots,r_m(x)$. 如果通过分析,能够知道 y 与 x 之间应该有什么样的函数关系,则 $r_1(x),\cdots,r_m(x)$ 容易确定. 若无法知道 y 与 x 之间的关系,通常可以将数据 (x_i,y_i), $i=1,2,\cdots,n$ 作出散点图(如图 7-2 所示),直观地判断应该用什么样的曲线去作拟合(如图7-3 所示). 人们常用的曲线有(参见图 7-4).

(1) 直线 $y=a_1x+a_2$.

(2) 多项式(一般 $m=2,3$,不宜过高) $y=a_1x^m+\cdots+a_mx+a_{m+1}$.

(3) 双曲线 $y=\dfrac{a}{bx+c}+d\ (a\neq0,b\neq0)$.

(4) 指数曲线 $y=ae^{bx}\quad(a\neq0,b\neq0)$.

(5) 对数曲线 $y=a\ln x+c\quad(a\neq0)$.

(6) 复合函数曲线 $y=\dfrac{a}{(bx+c)^2}\quad(a\neq0,b\neq0)$,

$$y=\frac{1}{ae^{bx}+c}\ (a\neq0,b\neq0),\ y=\frac{x}{ax+c}\ (a\neq0),\ y=axe^{bx}(a\neq0,b\neq0).$$

已知一组数据,用什么样的曲线拟合最好,可以在直观判断的基础上,选几种曲线分别作拟合,然后比较,看哪条曲线的最小二乘指标 J 最小(见(7.7)式).

例 7.3.1 给出一组实验数据点 (x_i,y_i) 的横坐标向量为 $\boldsymbol{x}=(-8.5,-8.7,-7.1,-6.8,-5.10,-4.5,-3.6,-3.4,-2.6,-2.5,-2.1,-1.5,-2.7,-3.6)$,纵横坐标向量为 $\boldsymbol{y}=(459.26,52.81,198.27,165.60,59.17,41.66,25.92,22.37,13.47,12.87,11.87,6.69,14.87,24.22)$,试用线性最小二乘法求拟合曲线,并用(7.2),(7.3)和(7.4)式估计其误差,作出拟合曲线.

解 (1) 首先根据给出的数据点 (x_i,y_i),用下列 MATLAB 程序画出散点图.

在 MATLAB 工作窗口输入程序

```
>>x =[ -8.5, -8.7, -7.1, -6.8, -5.10, -4.5, -3.6, -3.4, -2.6,
-2.5, -2.1, -1.5, -2.7, -3.6];
```

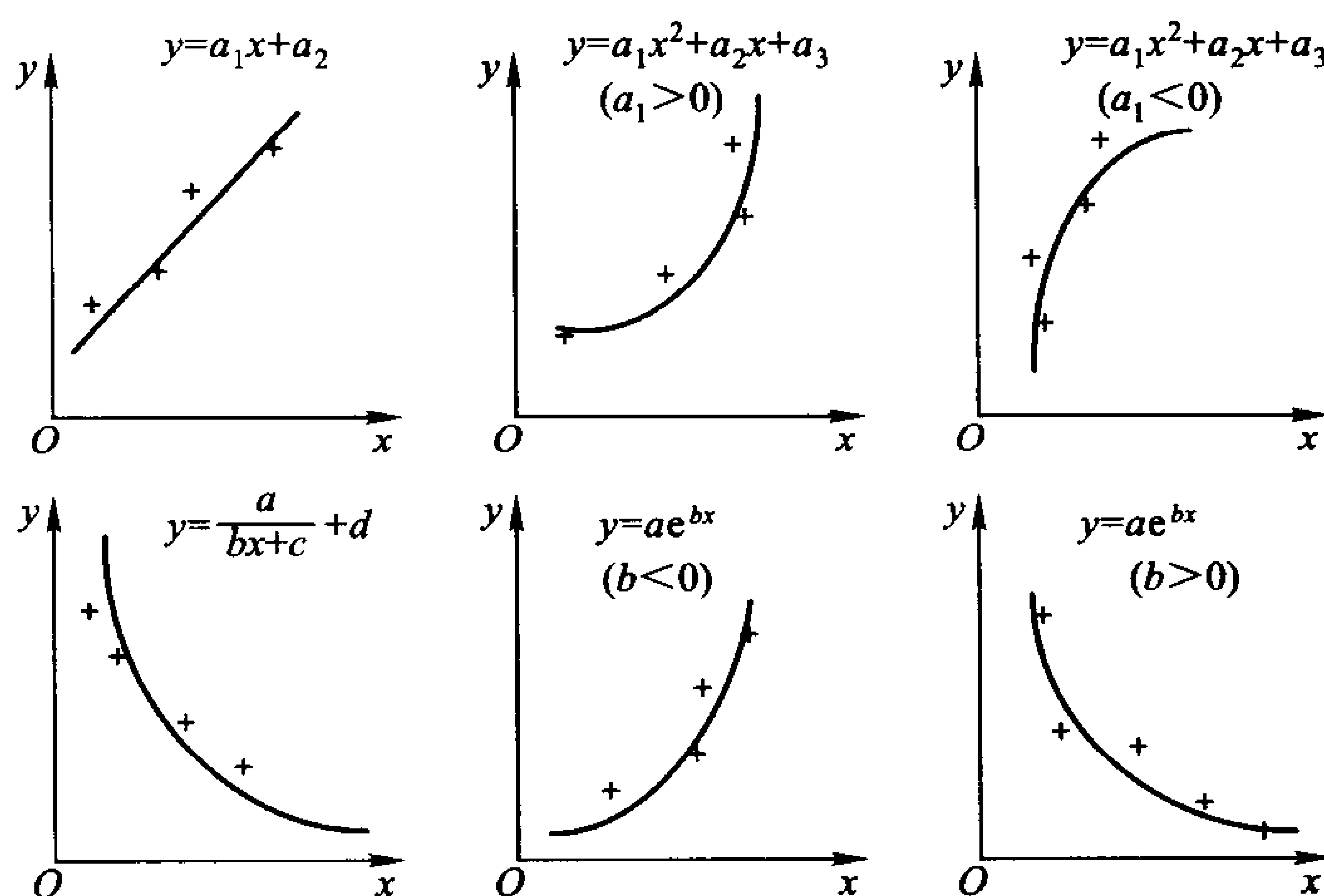


图 7-4 常用的拟合曲线

```
y=[459.26,52.81,198.27,165.60,59.17,41.66,25.92, 22.37,13.47,
12.87, 11.87,6.69,14.87,24.22];
plot(x,y,'r*'),legend('实验数据(xi,yi)')
xlabel('x'), ylabel('y'),
title('例7.3.1的数据点(xi,yi)的散点图')
```

运行后屏幕显示数据的散点图,见图 7-5(a).

(2) 因为数据的散点图 7-5(a)总的变化趋势与指数函数很接近,所以选取函数

$$f(x)=ae^{-bx}, \tag{7.13}$$

其中 a,b 是待定系数. 最好将奇异点(-8.7, 52.81)去掉.

(3) 用最小二乘准则求待定系数 a,b. 将给出的数据(x_i,y_i), $i=1,2,\cdots,14$ 代入(7.13)式,编写下列 MATLAB 程序计算$f(x)$在(x_i,y_i)处的函数值,即输入程序

```
>> syms a b
x=[-8.5, -8.7, -7.1, -6.8, -5.10, -4.5, -3.6, -3.4, -2.6, -2.5,
-2.1, -1.5, -2.7, -3.6];
fi=a.*exp(-b.*x)
```

运行后屏幕显示关于 a 和 b 的线性方程组

```
fi =
[ a*exp(17/2*b), a*exp(87/10*b), a*exp(71/10*b),a*
exp(34/5*b), a*exp(51/10*b), a*exp(9/2*b), a*exp(18/5*b), a*
```

```
exp(17/5*b), a*exp(13/5*b), a*exp(5/2*b), a*exp(21/10*b), a*
exp(3/2*b), a*exp(27/10*b), a*exp(18/5*b)]
```

根据(7.5)式,编写构造误差平方和 $J(a_1,\cdots,a_m)=\sum_{i=1}^{n}\delta_i^2=\sum_{i=1}^{n}[f(x_i)-y_i]^2$ 的 MATLAB 程序如下

```
>>y=[459.26,52.81,198.27,165.60,59.17,41.66,25.92,22.37,
13.47,12.87, 11.87, 6.69,14.87,24.22];
fi=[ a*exp(17/2*b),  a*exp(87/10*b),  a*exp(71/10*b),
a*exp(34/5*b),a*exp(51/10*b),  a*exp(9/2*b),  a*
exp(18/5*b),  a*exp(17/5*b),a*exp(13/5*b),  a*exp(5/2
*b),  a*exp(21/10*b),  a*exp(3/2*b),a*exp(27/10*b),  a
*exp(18/5*b)];
fy=fi-y;
fy2-fy.^2;
J=sum(fy.^2)
```

运行后屏幕显示误差平方和如下

```
J =
(a*exp(17/2*b)-22963/50)^2+(a*exp(87/10*b)-5281/100)^2
+(a*exp(71/10*b)-19827/100)^2+(a*exp(34/5*b)-828/5)^2+(a*exp
(51/10*b)-5917/100)^2+(a*exp(9/2*b)-2083/50)^2+(a*exp(18/5*
b)-648/25)^2+(a*exp(17/5*b)-2237/100)^2+(a*exp(13/5*b)-
1347/100)^2+(a*exp(5/2*b)-1287/100)^2+(a*exp(21/10*b)-1187/
100)^2+(a*exp(3/2*b)-669/100)^2+(a*exp(27/10*b)-1487/100)^2
+(a*exp(18/5*b)-1211/50)^2
```

为求 a,b 使 J 达到最小,只需利用极值的必要条件 $\frac{\partial J}{\partial a}=0,\frac{\partial J}{\partial b}=0$,得到关于 a,b 的线性方程组,这可以由下面的 MATLAB 程序完成,即输入程序

```
>> syms a b
J=(a*exp(17/2*b)-22963/50)^2+(a*exp(87/10*b)-5281/
100)^2+(a*exp(71/10*b)-19827/100)^2+(a*exp(34/5*b)-828/5)^2
+(a*exp(51/10*b)-5917/100)^2+(a*exp(9/2*b)-2083/50)^2+(a*
exp(18/5*b)-648/25)^2+(a*exp(17/5*b)-2237/100)^2+(a*exp(13/5
*b)-1347/100)^2+(a*exp(5/2*b) 1287/100)^2+(a*exp(21/10*b)-
1187/100)^2+(a*exp(3/2*b)-669/100)^2+(a*exp(27/10*b)-1487/
100)^2+(a*exp(18/5*b)-1211/50)^2;
```

```
Ja = diff(J,a);Jb = diff(J,b);
Ja1 = simple(Ja),Jb1 = simple(Jb),
```

运行后屏幕显示 J 分别对 a,b 的偏导数如下

```
Ja1 =
2*a*exp(3*b)+2*a*exp(17*b)+2*a*exp(87/5*b)+2*
exp(68/5*b)*a+2*exp(9*b)*a+2*a*exp(34/5*b)-669/50*exp(3/2
*b)-1487/50*exp(27/10*b)-2507/25*exp(18/5*b)-22963/25*exp
(17/2*b)-5281/50*exp(87/10*b)-19827/50*exp(71/10*b)-2237/50
*exp(17/5*b)-1656/5*exp(34/5*b)-1347/50*exp(13/5*b)-5917/50
*exp(51/10*b)-1287/50*exp(5/2*b)-2083/25*exp(9/2*b)-1187/50
*exp(21/10*b)+4*a*exp(36/5*b)+2*a*exp(26/5*b)+2*a*exp(71/
5*b)+2*a*exp(51/5*b)+2*a*exp(5*b)+2*a*exp(21/5*b)+2*a*
exp(27/5*b)
Jb1 =
1/500*a*(2100*a*exp(21/10*b)^2+8500*a*exp(17/2*b)^2
+6800*a*exp(34/5*b)^2-10035*exp(3/2*b)-40149*exp(27/10*b)-
180504*exp(18/5*b)-3903710*exp(17/2*b)-459447*exp(87/10*b)-
1407717*exp(71/10*b)-76058*exp(17/5*b)-1126080*exp(34/5*b)-
35022*exp(13/5*b)-301767*exp(51/10*b)-32175*exp(5/2*b)-
187470*exp(9/2*b)-24927*exp(21/10*b)+7100*a*exp(71/10*b)^2+
5100*a*exp(51/10*b)^2+4500*a*exp(9/2*b)^2+7200*a*exp(18/5*
b)^2+3400*a*exp(17/5*b)^2+2600*a*exp(13/5*b)^2+2500*a*
exp(5/2*b)^2+1500*a*exp(3/2*b)^2+2700*a*exp(27/10*b)^2+8700
*a*exp(87/10*b)^2)
```

用解二元非线性方程组的牛顿法的 MATLAB 程序求解线性方程组 $J_{a1}=0$, $J_{b1}=0$,得

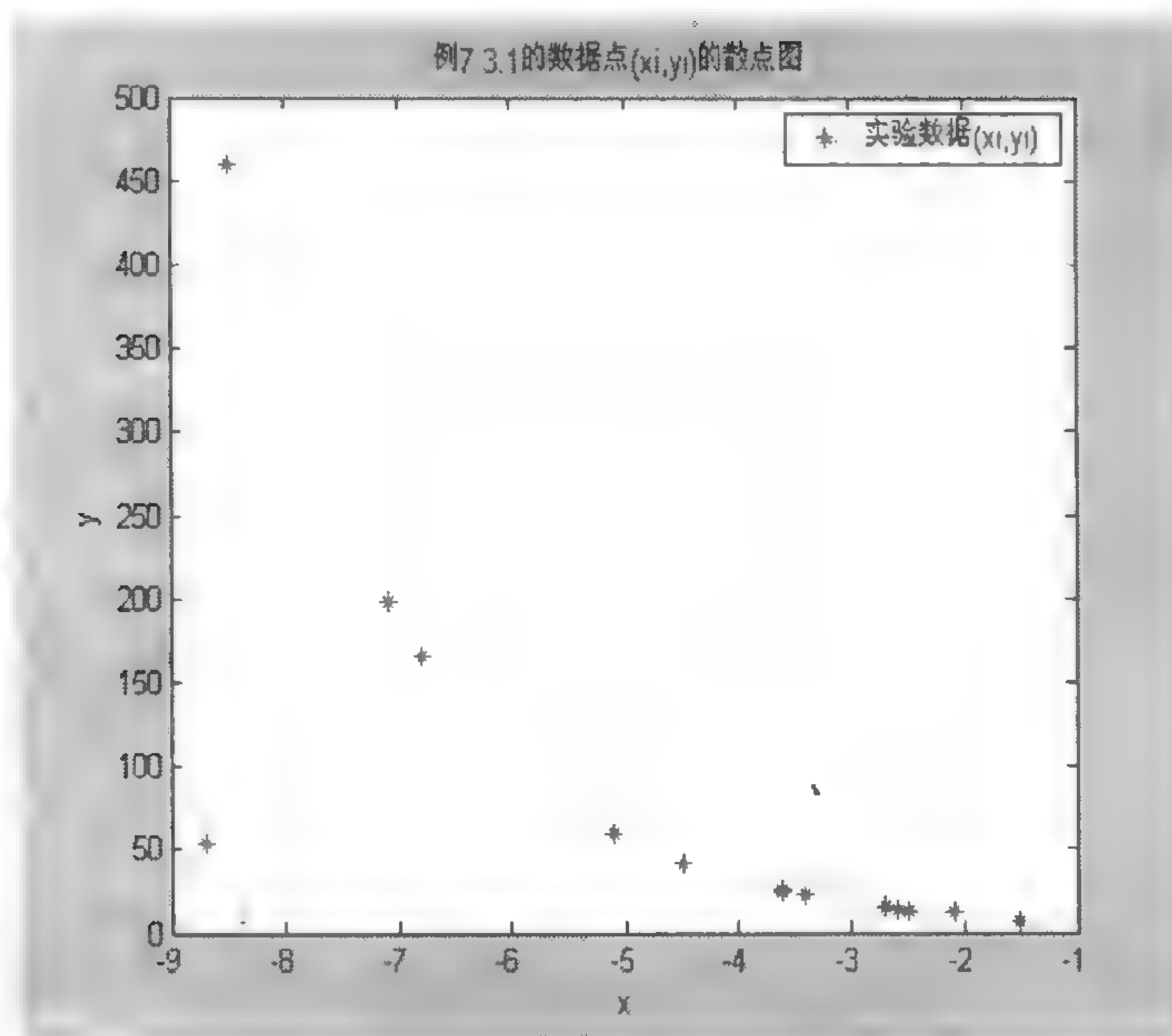
```
a =            b =
   2.8110        0.5816
```

故所求的拟合曲线(7.13)为

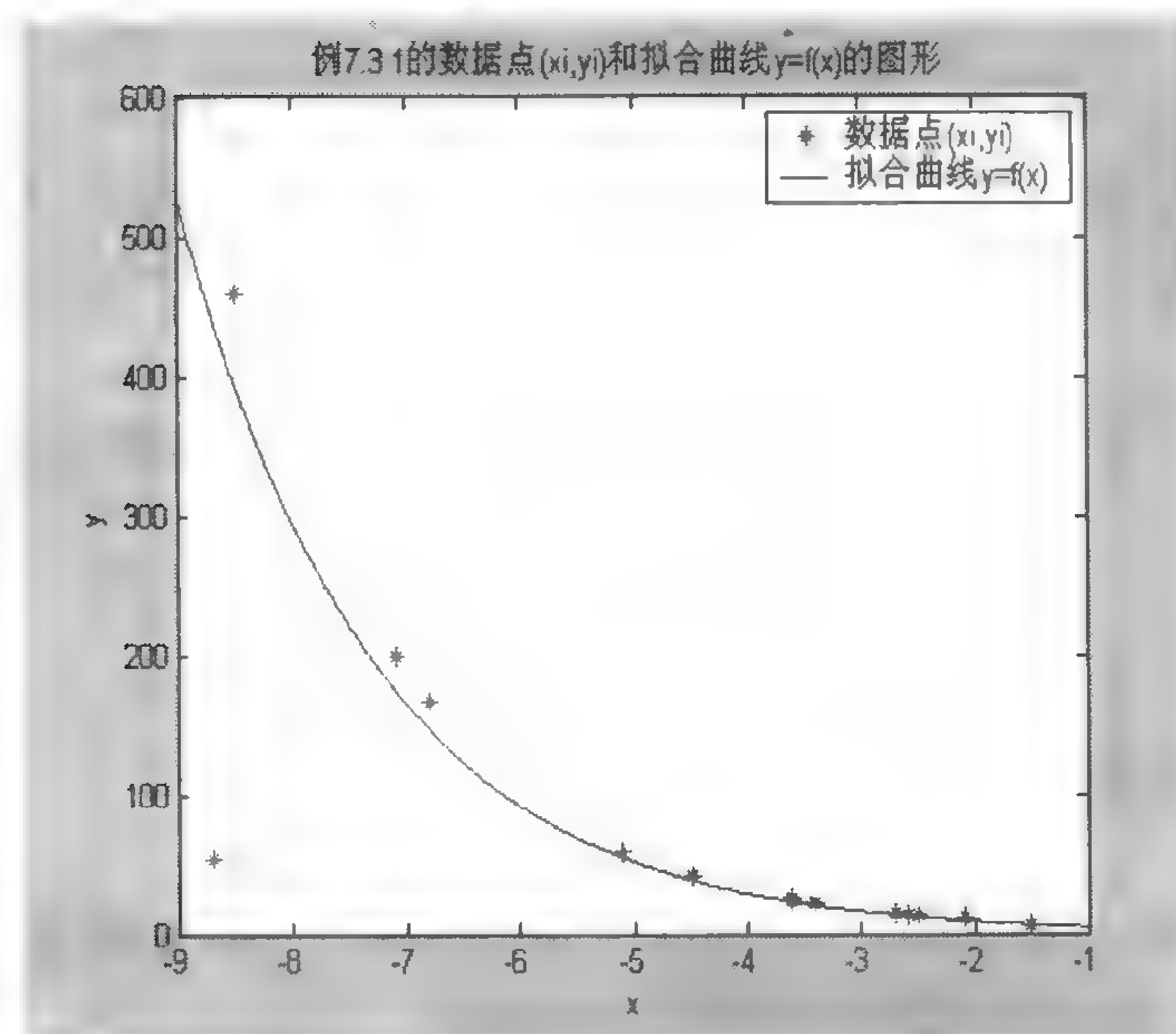
$$f(x)=2.8110e^{-0.5816x} \tag{7.14}$$

(4) 根据(7.2),(7.3),(7.4)和(7.14)式编写下面的 MATLAB 程序估计其误差,并作出拟合曲线和数据的图形. 输入程序

```
>> xi =[-8.5 -8.7  -7.1  -6.8  -5.10  -4.5 -3.6 -3.4 -2.6
-2.5 -2.1  -1.5  -2.7  -3.6];
y =[459.26  52.81  198.27  165.60  59.17  41.66  25.92 22.37
13.47  12.87  11.87  6.69  14.87  24.22];
```



(a) 例 7.3.1 数据的散点图

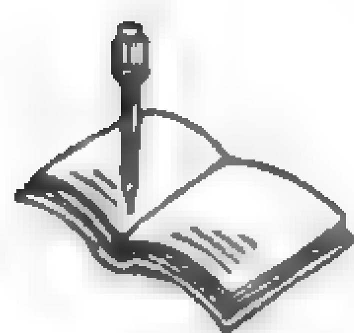


(b) 数据散点图和拟合曲线的图形

图 7-5

```
n=length(xi); f=2.8110.*exp(-0.5816.*xi); x=-9:0.01:-1;
F=2.8110.*exp(-0.5816.*x); fy=abs(f-y); fy2=fy.^2; Ew=
max(fy),
E1=sum(fy)/n, E2=sqrt((sum(fy2))/n), plot(xi,y,'r*'), hold on
plot(x,F,'b-'), hold off,
legend('数据点(xi,yi)','拟合曲线 y=f(x)')
xlabel('x'), ylabel('y'),
title('例 7.3.1 的数据点(xi,yi)和拟合曲线 y=f(x)的图形')
```

运行后屏幕显示数据(x_i,y_i)与拟合函数f的最大误差$E_w=390.1415$,平均误差$E_1=36.9422$和均方根误差$E_2=106.0317$及其数据点(x_i,y_i)和拟合曲线$y=f(x)$的图形,见图 7-5(b).



习 题 7.3

1. 给出一组实验数据点(x_i,y_i)的横坐标向量为$x=(-7.4\quad -6.8\quad -5.16\quad -4.5\quad -3.6\quad -3.4\quad -2.6\quad -2.5\quad -2.4\quad -1.5\quad -2.7\quad -3.6)$,纵坐标向量为$y=(198.27\quad 165.66\quad 59.17\quad 41.66\quad 25.92\quad 22.37\quad 13.47\quad 12.87\quad 11.87\quad 6.69\quad 14.87\quad 24.22)$,试用线性最小二乘法求拟合曲线,并用(7.2),(7.3)和(7.4)式估计其误差,作出拟合曲线.

2. 给出下面表中的一组数据点(x_i,y_i),试用线性最小二乘法求拟合曲线,并用(7.2),(7.3)和(7.4)式估计其误差,作出拟合曲线.

x_i	-1.7	-1.1	-0.8	0	0.1	1.5	2.7	3.6
y_i	-85.50	-36.15	-26.52	-9.10	-8.43	-13.12	6.50	68.04

3. 在化工生产中常常需要知道丙烷在各种温度T和压力p下的导热系数K. 下面是实验得到的一组数据:

T(℃)	p(10^3kN/m^2)	K	T(℃)	p(10^3kN/m^2)	K
68	9.798	0.085	106	9.792	0.070
68	13.324	0.090	106	14.277	0.075
87	9.008	0.076	140	9.656	0.061
87	13.355	0.081	140	12.463	0.065

试求$T=99$ ℃和$p=10.3\times10^3$ kN/m^{2}下的K.

7.4 多项式拟合及其 MATLAB 程序

面对一组数据(x_i,y_i),$i=1,2,\cdots,n$,用线性最小二乘法作曲线拟合时,如果

选取一组函数 $r_1(x),\cdots,r_m(x)$ 为 $1,x,x^2,\cdots,x^m(m<n)$，则拟合曲线为多项式

$$y=a_1x^m+\cdots+a_mx+a_{m+1} \tag{7.15}$$

一般 $m=2,3$，不宜过高.

对于指数曲线，拟合前需作变量代换，化为系数参数的线性函数.

用 MATLAB 作线性最小二乘拟合的多项式拟合有现成程序，调用格式为

```
a=polyfit(x,y,m)
```

其中输入参数 x,y 为要拟合的数据，是长度自定义的数组，m 为拟合多项式的次数，输出参数 a 为拟合多项式 $y=a_1x^m+\cdots+a_mx+a_{m+1}$ 的系数向量 $\boldsymbol{a}=(a_1,\cdots,a_m,a_{m+1})$.

多项式在 x 处的值 y 可用下面程序计算

```
y=polyval(a,x)
```

线性最小二乘拟合的另一种作法是，用 MATLAB 直接解超定方程组(方程个数大于未知数个数).

对于选定的 $r_k(x),k=1,\cdots,m$ 和已知数据 $(x_i,y_i),i=1,2,\cdots,n\ (m<n)$，对(7.15)式令 $f(x_i)=y_i$，得 n 个方程、m 个未知数的**超定方程组**

$$\boldsymbol{RA}=\boldsymbol{y}. \tag{7.16}$$

这个方程当然没有普通意义下的解，但是可以在最小二乘意义下求解. 在用 MATLAB 求解线性代数方程组 $\boldsymbol{RA}=\boldsymbol{y}$ 时，我们知道，若输入 $n\times n$ 可逆方阵 $\boldsymbol{R}$ 和 n 维向量 $\boldsymbol{y}$，则键入 A=R\Y，运行后得到普通意义下的解 $\boldsymbol{A}$.

这里要指出的是，MATLAB 具有如下功能：若输入 $n\times m$ 阵 $\boldsymbol{R}$ 和 n 维向量 $\boldsymbol{y}$ $(m<n)$，只要 $\boldsymbol{R}^T\boldsymbol{R}$ 可逆，仍键入 A=R\Y，则给出最小二乘准则下的解 $\boldsymbol{A}$.

例 7.4.1 给出一组数据点 (x_i,y_i) 列入表 7-3 中，试用线性最小二乘法求拟合曲线，并用(7.2)，(7.3)和(7.4)式估计其误差，作出拟合曲线.

表 7-3 例 7.4.1 的一组数据 (x_i,y_i)

x_i	-2.9	-1.9	-1.1	-0.8	0	0.1	1.5	2.7	3.6
y_i	53.94	33.68	20.88	16.92	8.79	8.98	4.17	9.12	19.88

解 (1) 首先根据表 7-3 给出的数据点 (x_i,y_i)，用下列 MATLAB 程序画出散点图.

在 MATLAB 工作窗口输入程序

```
>> x=[-2.9  -1.9  -1.1  -0.8  0  0.1  1.5  2.7  3.6];
y=[53.94  33.68  20.88  16.92  8.79  8.98  4.17  9.12  19.88];
plot(x,y,'r*'), legend('数据点(xi,yi)')
xlabel('x'), ylabel('y'),
title('例7.4.1 的数据点(xi,yi)的散点图')
```

运行后屏幕显示数据的散点图,见图 7－6(a).

(2) 因为数据的散点图 7－6(a)的变化趋势与二次多项式很接近,所以选取一组函数 $1,x,x^2$,令

$$f(x)=a_1x^2+a_2x+a_3, \tag{7.17}$$

其中 a_k 是待定系数($k=1,2,3$).

(3) 用作线性最小二乘拟合的多项式拟合的 MATLAB 程序求待定系数 a_k ($k=1,2,3$). 输入程序

```
>> a =polyfit(x,y,2)
```

运行后输出(7.17)式的系数

```
a =
   2.8302     -7.3721     9.1382
```

故拟合多项式为

$$f(x)=2.830\,2x^2-7.372\,1x+9.138\,2. \tag{7.18}$$

(4) 根据(7.2),(7.3),(7.4)和(7.18)式编写下面的 MATLAB 程序估计其误差,并作出拟合曲线和数据的图形. 输入程序

```
>> xi =[ -2.9  -1.9  -1.1  -0.8  0  0.1  1.5  2.7  3.6];
y =[53.94  33.68  20.88  16.92  8.79  8.98  4.17  9.12  19.88];
n = length(xi); f =2.8302.*xi.^2 -7.3721.*xi +9.1382
x = -2.9:0.001:3.6;F =2.8302.*x.^2 -7.3721.*x +9.1382;
fy =abs(f -y); fy2 =fy.^2; Ew =max(fy), E1 =sum(fy)/n,
E2 =sqrt((sum(fy2))/n), plot(xi,y,'r*', x,F,'b -'),
legend('数据点(xi,yi)','拟合曲线 y = f(x)')
xlabel('x'), ylabel('y'),
title('例 7.4.1 的数据点(xi,yi)和拟合曲线 y = f(x)的图形')
```

运行后屏幕显示数据(x_i,y_i)与拟合函数 f 的最大误差 E_w,平均误差 E_1 和均方根误差 E_2 及其数据点(x_i,y_i)和拟合曲线 $y=f(x)$的图形,见图 7－6(b).

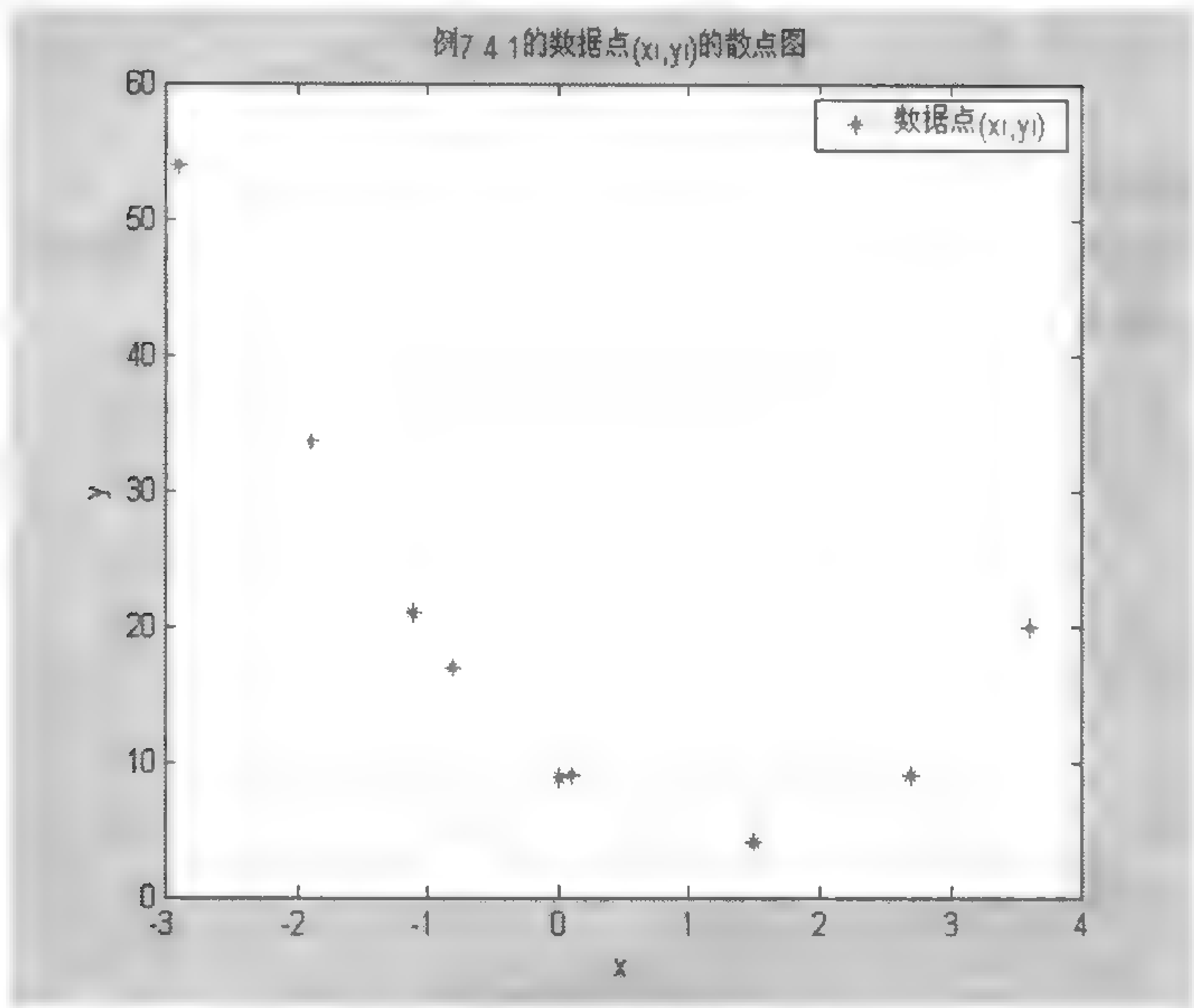
```
Ew =            E1 =            E2 =
      0.745 7         0.389 2         0.436 3
```

例 7.4.2 有一只对温度敏感的电阻,已经测得了一组温度 T 和电阻 R 数据(T_i,r_i) (见表 7－4):

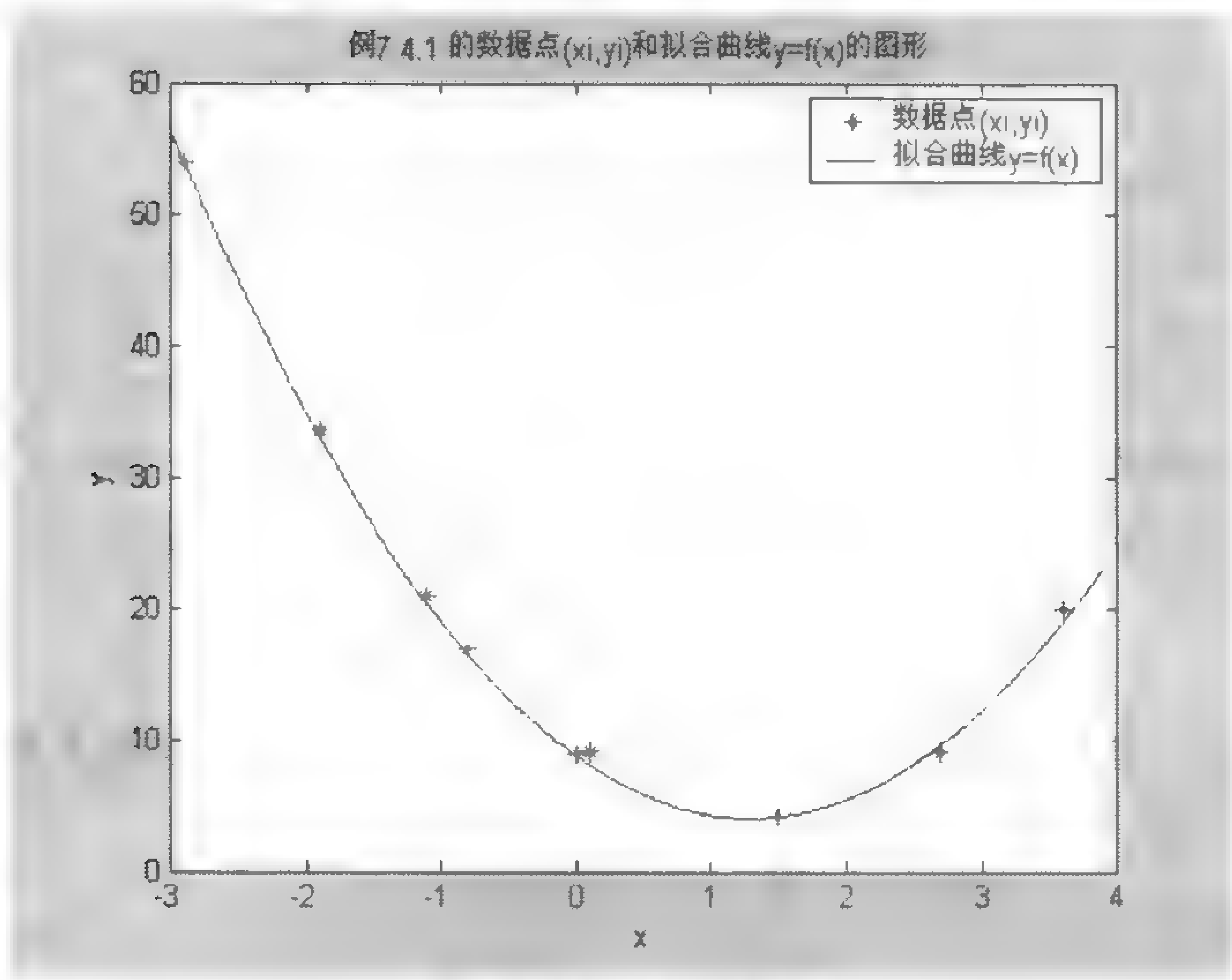
表 7－4 例 7.4.2 的一组数据(T_i,r_i)

T(℃)	20.5	32.5	51.0	73.0	95.7
R(Ω)	765	826	873	942	1 032

现在想知道 60 ℃时的电阻多大. 用上述两种方法拟合电阻 R 与温度 T 之间的



(a) 表 7-3 给出的数据的散点图



(b) 数据散点图和拟合曲线

图 7-6

关系,并估计三种误差.

解 方法1 (1) 输入 MATLAB 程序

```
>> t =[20.5 32.5 51 73 95.7]; r =[765 826 873 942 1032];
plot(t,r,'r*'), legend('数据点(ti,ri)')
xlabel('t'), ylabel('r'),
title('例7.4.2 的数据点(ti,ri)的散点图')
```

运行后得到表 7-3 给出的数据的散点图 7-7,图中 * 号表示的 5 个数据点.

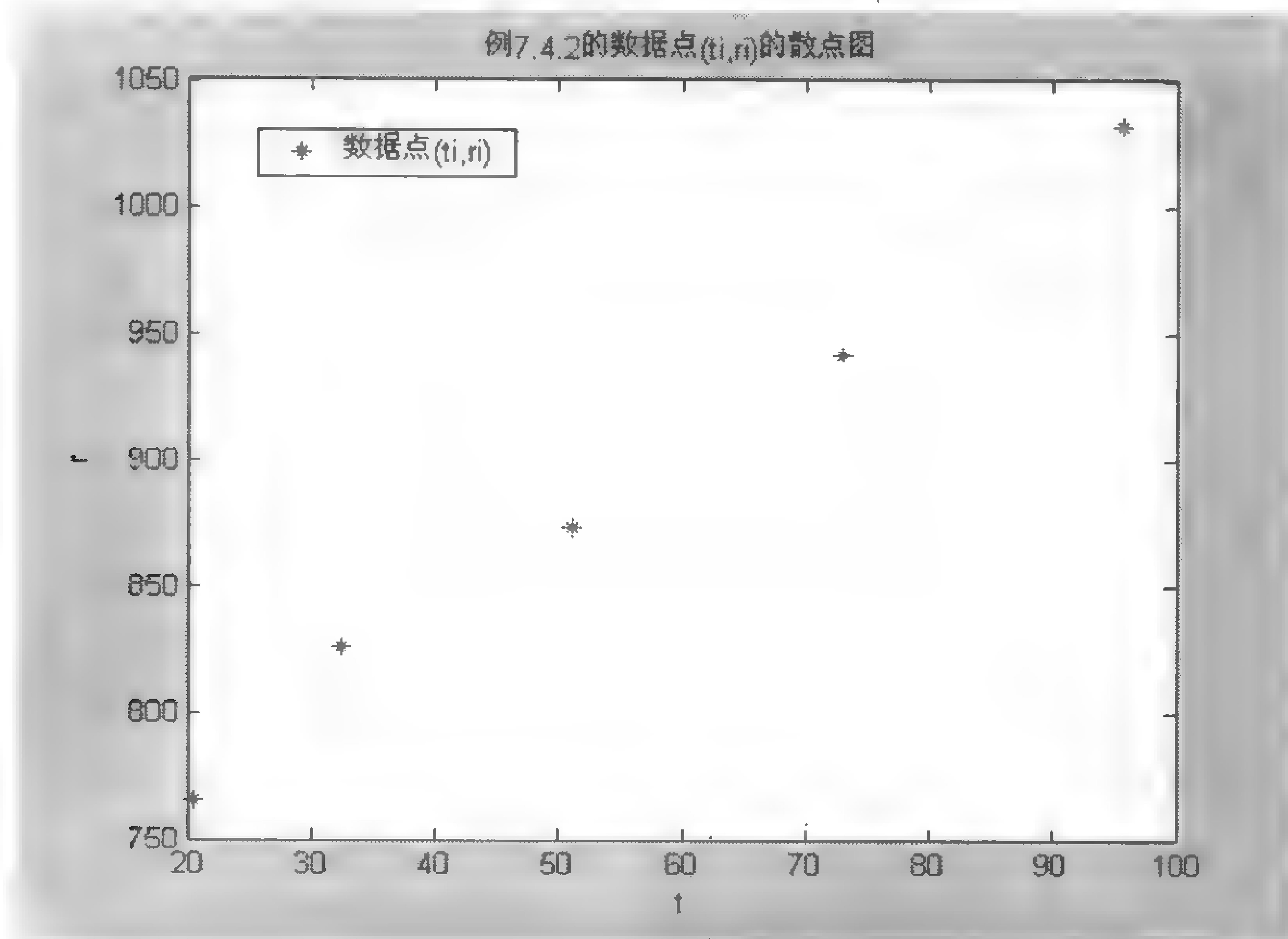


图 7-7 电阻 R 与温度 T 之间的关系

(2) 由图 7-7 可见,5 个数据点基本在一条直线上,所以,拟合曲线选择直线 $R=aT+b$,输入如下程序

```
>> t =[20.5 32.5 51 73 95.7]; r =[765 826 873 942 1032];
n = length(t); f = polyfit(t,r,1); a = f(1), b = f(2),
y - polyval(f,t);
plot(t,r,'r*',t,y,'b-'), xlabel('t'),ylabel('R')
legend('数据点(ti,ri)','拟合直线 R = f(t)')
title('例7.4.2 的数据点(ti,ri)和拟合直线 R = f(t)的图形')
fy = abs(y - r); fy2 = fy.^2; Ew = max(fy), E1 = sum(fy)/n,
E2 = sqrt((sum(fy2))/n)
```

运行后屏幕显示拟合直线 $R=aT+b$ 的系数 $a=3.394\,0$, $b=702.491\,8$, 数据(t_i, r_i)与拟合函数 f 的最大误差 $E_w=13.203\,5$, 平均误差 $E_1=7.162\,8$ 和均方根误

差 $E_2=8.0152$ 及其数据点(t_i,r_i)和拟合直线 $R=f(T)$的图形(见图 7-8).

方法 2 输入如下程序

```
>> t=[20.5 32.5 51 73 95.7];
r=[765 826 873 942 1032]; R=[t' ones(5,1)]; aa=R\r';
a=aa(1), b=aa(2), y=polyval(aa,t);
plot(t,r,'k+',t,y,'r'), xlabel('t'), ylabel('R')
```

得到完全相同的结果.

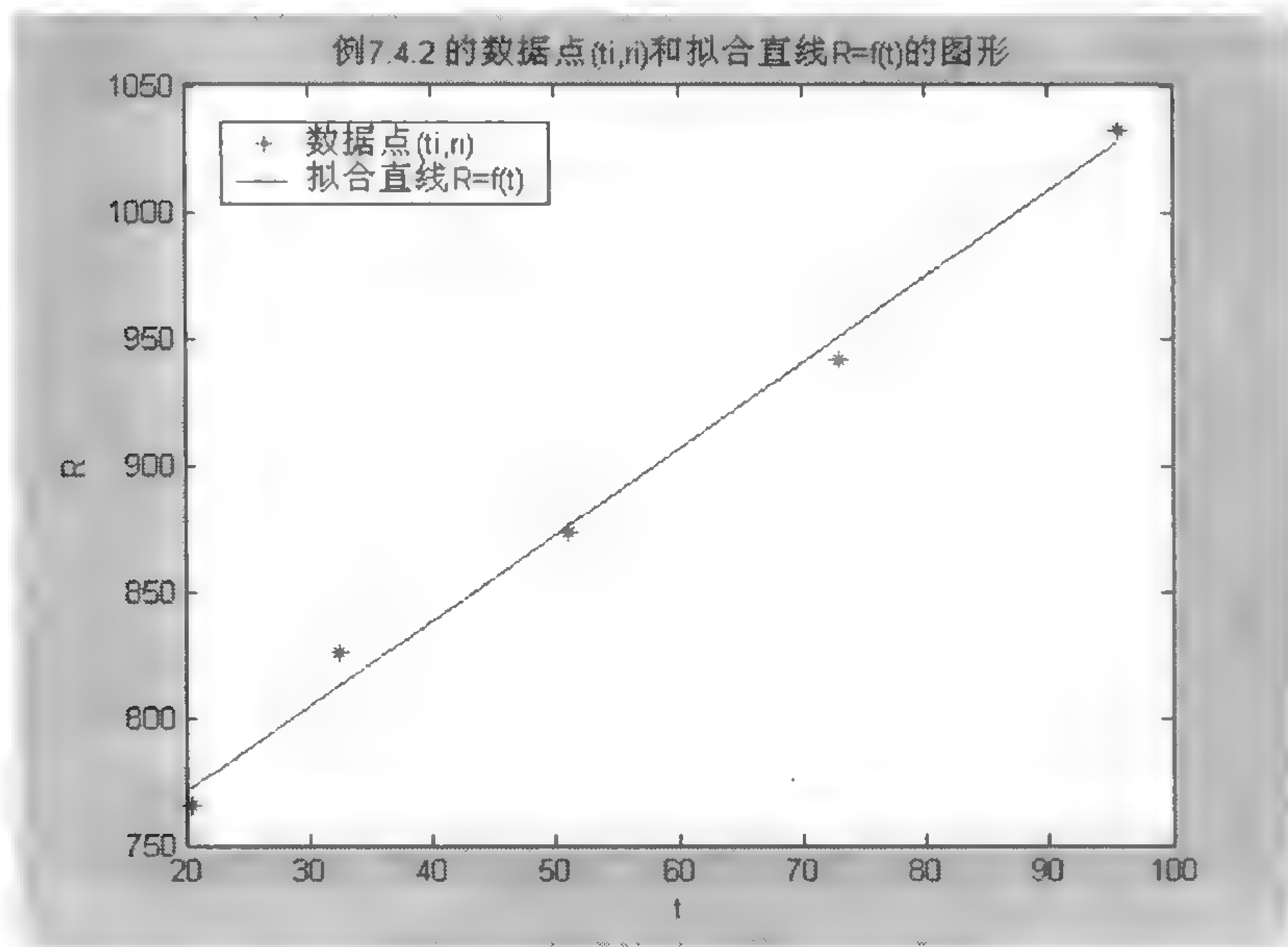
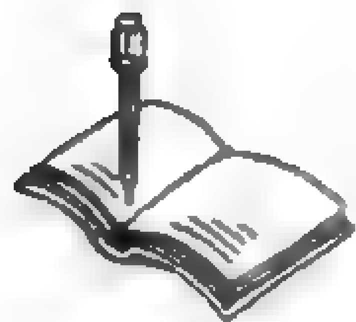


图 7-8 数据点(t_i,r_i)和拟合直线



习 题 7.4

1. 给定的多项式 $y=x^3-6x^2+5x-3$,如果产生一组数据(x_i,y_i),$i=1,2,\cdots,n$,再在 y_i 上添加随机干扰(可用 rand 产生(0,1)均匀分布随机数,或用 randn 产生 $N(0,1)$分布随机数),然后用 x_i 和添加了随机干扰 y_i 作 3 次多项式拟合,与原系数比较. 如果作 2 或 4 次多项式拟合,结果如何?

2. 自己实测一组数据(或利用给出的数据),确定模型 $t=an^2+bn$ 中的系数 a,b.

7.5 拟合曲线的线性变换及其 MATLAB 程序

从例 7.3.1 的计算可以发现，用指数函数作拟合曲线时，计算复杂，而且初始值如果选择得不适当，迭代序列不能够保证收敛到要求的系数参数. 为了简化计算，尽量不用迭代法，人们经常采取数据线性化技术来拟合各种曲线. 例如，将 $y=ae^{-bx}(a>0)$ 两端取对数，化为 $\ln y=\ln a-bx$，令 $Y=\ln y, A=\ln a, B=-b$，则指数函数变换为线性函数 $Y=A+Bx$，用多项式拟合的命令 f = polyfit(x, Y,1)，可以求出 A,B，从而得到 a,b 的值. 常用的线性变换如表 7－5 所示.

表 7－5 常用的线性变换

函数 $y=f(x)$	化为线性函数 $Y=AX+B$ 型	变量与常量的变化
$y=ae^{-bx}\ (a>0,b\neq0)$	$\ln y=\ln a-bx$	$Y=\ln y, B=\ln a, A=-b, X=x$
$y=ae^{bx}\ (a>0,b\neq0)$	$\ln y=\ln a+bx$	$Y=\ln y, B=\ln a, A=b, X=x$
$y=\dfrac{a}{x}+b\ (a\neq0)$	$y=a\dfrac{1}{x}+b$	$A=a, X=\dfrac{1}{x}, B=b, Y=y$
$y=\dfrac{a}{bx+c}\ (ab\neq0)$	$\dfrac{1}{y}=\dfrac{b}{a}x+\dfrac{c}{a}$	$Y=\dfrac{1}{y}, A=\dfrac{b}{a}, B=\dfrac{c}{a}, X=x$
$y=\dfrac{x}{ax+b}\ (a\neq0)$	$\dfrac{1}{y}=b\dfrac{1}{x}+a$	$Y=\dfrac{1}{y}, X=\dfrac{1}{x}, B=a, A=b$
$y=\dfrac{1}{(ax+b)^2}\ (a\neq0)$	$y^{-\frac{1}{2}}=ax+b$	$Y=y^{-\frac{1}{2}}, X=x, A=a, B=b$
$y=\dfrac{c}{ae^{bx}+1}\ (abc\neq0)$	$\ln(\dfrac{c}{y}-1)=\ln a+bx$	$Y=\ln(\dfrac{c}{y}-1), B=\ln a, A=b$
$y=axe^{bx}(a>0)$	$\ln\dfrac{y}{x}=\ln a+bx$	$Y=\ln\dfrac{y}{x}, B=\ln a, A=b, X=x$

例 7.5.1 给出一组实验数据点 (x_i, y_i) 的横坐标向量为 $\boldsymbol{x}=(7.5\ \ 6.8\ \ 5.10\ \ 4.5\ \ 3.6\ \ 3.4\ \ 2.6\ \ 2.5\ \ 2.1\ \ 1.5\ \ 2.7\ \ 3.6)$，纵横坐标向量为 $\boldsymbol{y}=(359.26\ \ 165.60\ \ 59.17\ \ 41.66\ \ 25.92\ \ 22.37\ \ 13.47\ \ 12.87\ \ 11.87\ \ 6.69\ \ 14.87\ \ 24.22)$，试用线性变换和线性最小二乘法求拟合曲线，并用(7.2)，(7.3)和(7.4)式估计其误差，作出拟合曲线.

解 (1)首先根据给出的数据点 (x_i, y_i)，用下列 MATLAB 程序画出散点图.

在 MATLAB 工作窗口输入程序

```
>> x =[7.5  6.8  5.10  4.5  3.6  3.4  2.6  2.5  2.1  1.5  2.7  3.6];
y =[359.26  165.60  59.17  41.66  25.92  22.37  13.47  12.87
```

```
11.87  6.69  14.87   24.22];
        plot(x,y,'r*'), legend('数据点(xi,yi)')
        xlabel('x'), ylabel('y'),
        title('例7.5.1 的数据点(xi,yi)的散点图')
```

运行后屏幕显示数据的散点图,见图 7-9.

(2) 根据数据散点图 7-9,取拟合曲线为

$$y = ae^{bx}(a>0, b\neq 0), \tag{7.19}$$

其中 a,b 是待定系数. 令 $Y=\ln y, A=\ln a, B=b$,则(7.19)化为 $Y=A+Bx$. 在 MATLAB 工作窗口输入程序

```
        >> x =[7.5  6.8  5.10  4.5  3.6  3.4  2.6  2.5  2.1  1.5  2.7  3.6];
        y =[359.26  165.60  59.17  41.66  25.92  22.37  13.47  12.87
11.87  6.69  14.87   24.22];
        Y =log(y); a =polyfit(x,Y,1); B =a(1);A =a(2); b =B,a =exp(A)
        n =length(x); X =8: -0.01:1; Y =a * exp(b.* X); f =a * exp(b.* x);
        plot(x,y,'r*',X,Y,'b-'), xlabel('x'),ylabel('y')
        legend('数据点(xi,yi)','拟合曲线 y =f(x)')
        title('例7.5.1 的数据点(xi,yi)和拟合曲线 y - f(x)的图形')
        fy =abs(f -y); fy2 =fy.^2;
        Ew =max(fy), E1 =sum(fy)/n, E2 =sqrt((sum(fy2))/n)
```

运行后屏幕显示 $y=ae^{bx}$的系数 $b=0.624\ 1, a=2.703\ 9$,数据(x_i,y_i)与拟合函数f的最大误差 $E_w=67.641\ 9$,平均误差 $E_1=8.677\ 6$ 和均方根误差 $E_2=20.711\ 3$及其数据点(x_i,y_i)和拟合曲线 $f(x)=2.703\ 9e^{0.624\ 1x}$的图形,见图 7-10.

例 7.5.2(给药方案) 一种新药用于临床之前,必须设计给药方案. 在快速静脉注射的给药方式下,所谓给药方案是指,每次注射剂量多大,间隔时间多长. 药物进入机体后随血液输送到全身,在这个过程中不断地被吸收、分布、代谢,最终排出体外. 药物在血液中的浓度,即单位体积血液中的药物含量,称血药浓度. 在最简单的一室模型中,将整个机体看作一个房室,称中心室,室内的血药浓度是均匀的. 快速静脉注射后,浓度立即上升;然后逐渐下降. 当浓度太低时,达不到预期的治疗效果;血药浓度太高,又可能导致药物中毒或副作用太强. 临床上,每种药物有一个最小有效浓度 c_1 和一个最大治疗浓度 c_2. 设计给药方案时,要使血药浓度保持在 $c_1\sim c_2$ 之间. 设本题所研究药物的最小有效浓度 $c_1=10\ \mu g/mL$,最大治疗浓度 $c_2=25\ \mu g/mL$. 在实验中,对某人用快速静脉注射方式一次注入该药物 300 mg 后,在一定时刻 t h 采集血样,测得血药浓度 $c\ \mu g/mL$ 如表7-6. 求给药后血药浓度随时间变化的规律,估计误差,并设计给药方案.

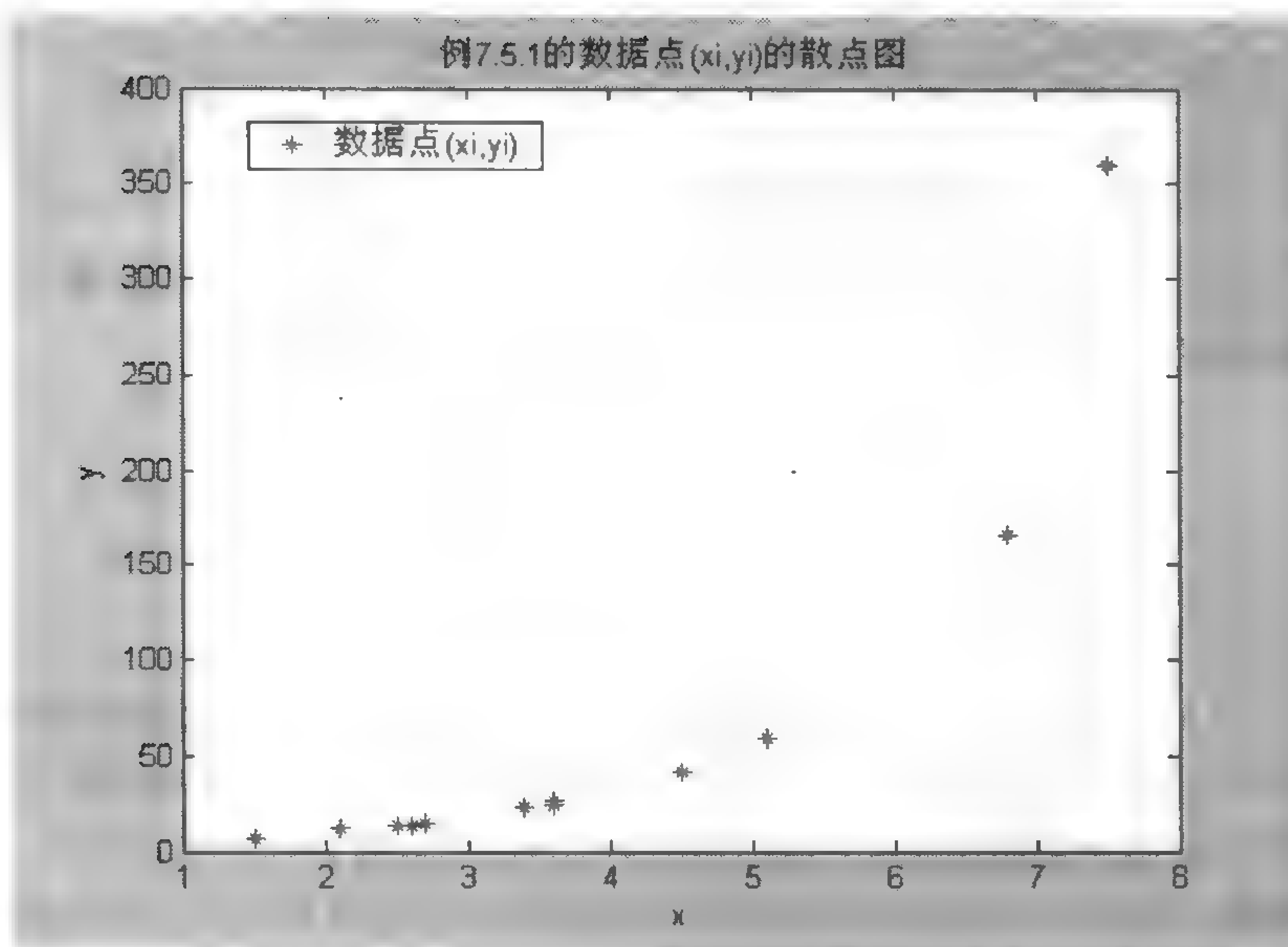


图 7－9　例 7.5.1 数据点的散点图

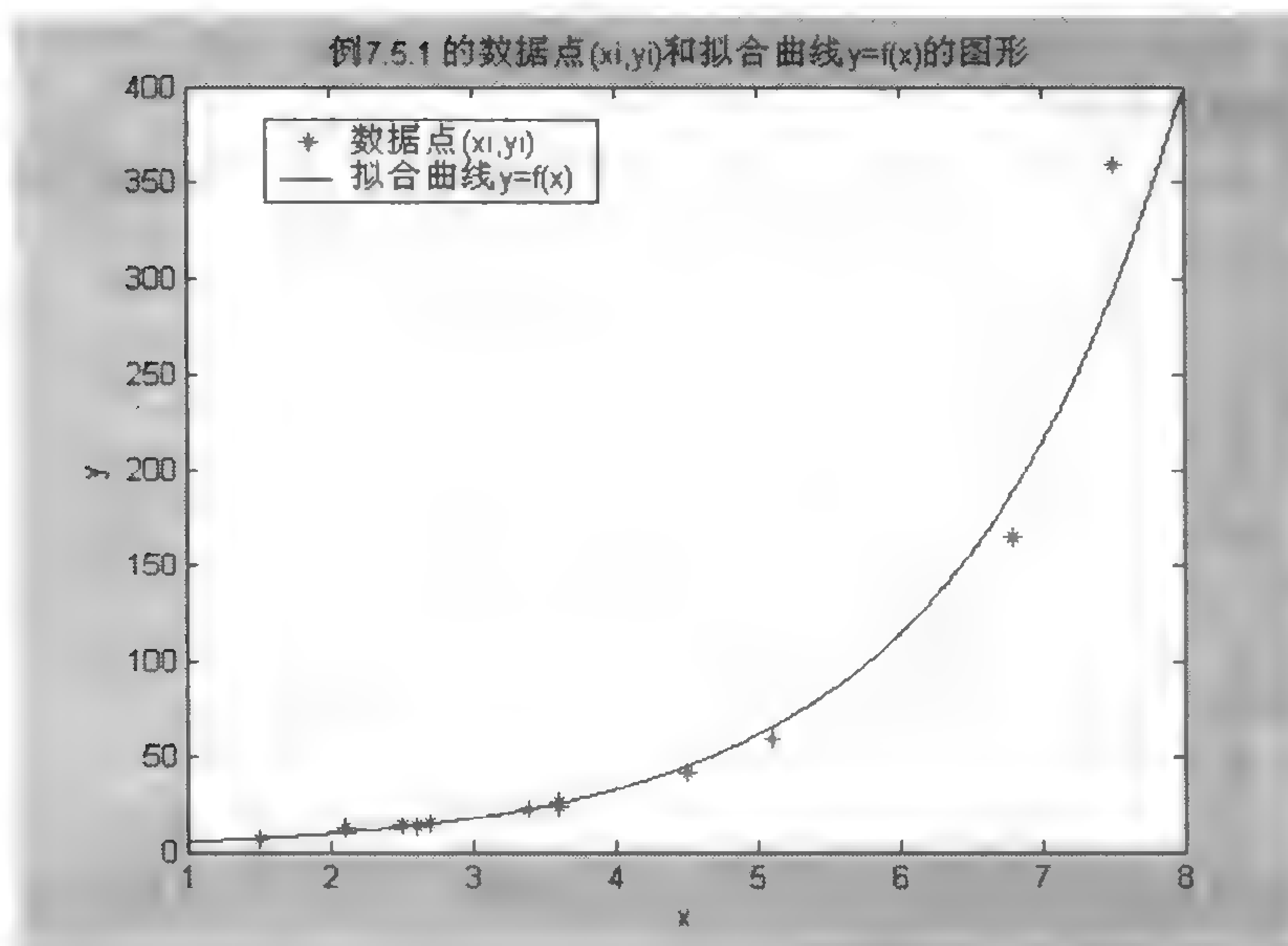


图 7－10　散点图和拟合曲线

表 7-6　例 7.5.2 的血药浓度 $c(t)$ 的测试数据

t	0.25	0.5	1	1.5	2	3	4	6	8
c	19.21	18.15	15.36	14.10	12.89	9.32	7.45	5.24	3.01

解　(1) 从表 7-6 可以看出,随着 t 的增加,c 并非按线性规律减少. 为了找出较准确的规律,画数据的散点图,输入 MATLAB 程序

```
>> t =[0.25  0.5  1  1.5  2  3  4  6  8];
c =[19.21 18.15 15.36 14.10 12.89 9.32 7.45 5.24 3.01];
plot(t,c,'r*')
xlabel('t'),ylabel('C'), legend('数据点(ti,ci)')
title('例7.5.2 的数据点(ti,ci)的图形')
```

得图 7-11(a),可知 t 与 c 近似指数函数关系,即 $c(t)$ 有按负指数规律减少的趋势. 在理论方面,若某瞬时注入一定剂量的药物,设中心室血液的容积是常数,对血药浓度变化规律最简单的、又是合理的假设为:药物向体外排除的速率与中心室的血药浓度成正比. 由此可以建立血药浓度 $c(t)$ 的微分方程,然后求解. 最后,根据理论分析的结果和实验数据设计给药方案.

(2) 确定血药浓度的变化规律.

根据上述给药方案中的分析,作如下简化假设:

(i) 药物向体外排除的速率与中心室的血药浓度成正比,比例系数为 k (>0),**称排除速率**;

(ii) 中心室血液容积为常数 V,$t=0$ 瞬时注入药物的剂量为 d,则血药浓度为 d/V. 由假设(i),中心室的血药浓度 $c(t)$ 应满足微分方程

$$\frac{\mathrm{d}c}{\mathrm{d}t}=-kc. \tag{7.20}$$

由假设(ii),方程的初始条件为

$$c(0)=d/V. \tag{7.21}$$

求解(7.19)可得

$$c(t)=\frac{d}{V}\mathrm{e}^{-kt}, \tag{7.22}$$

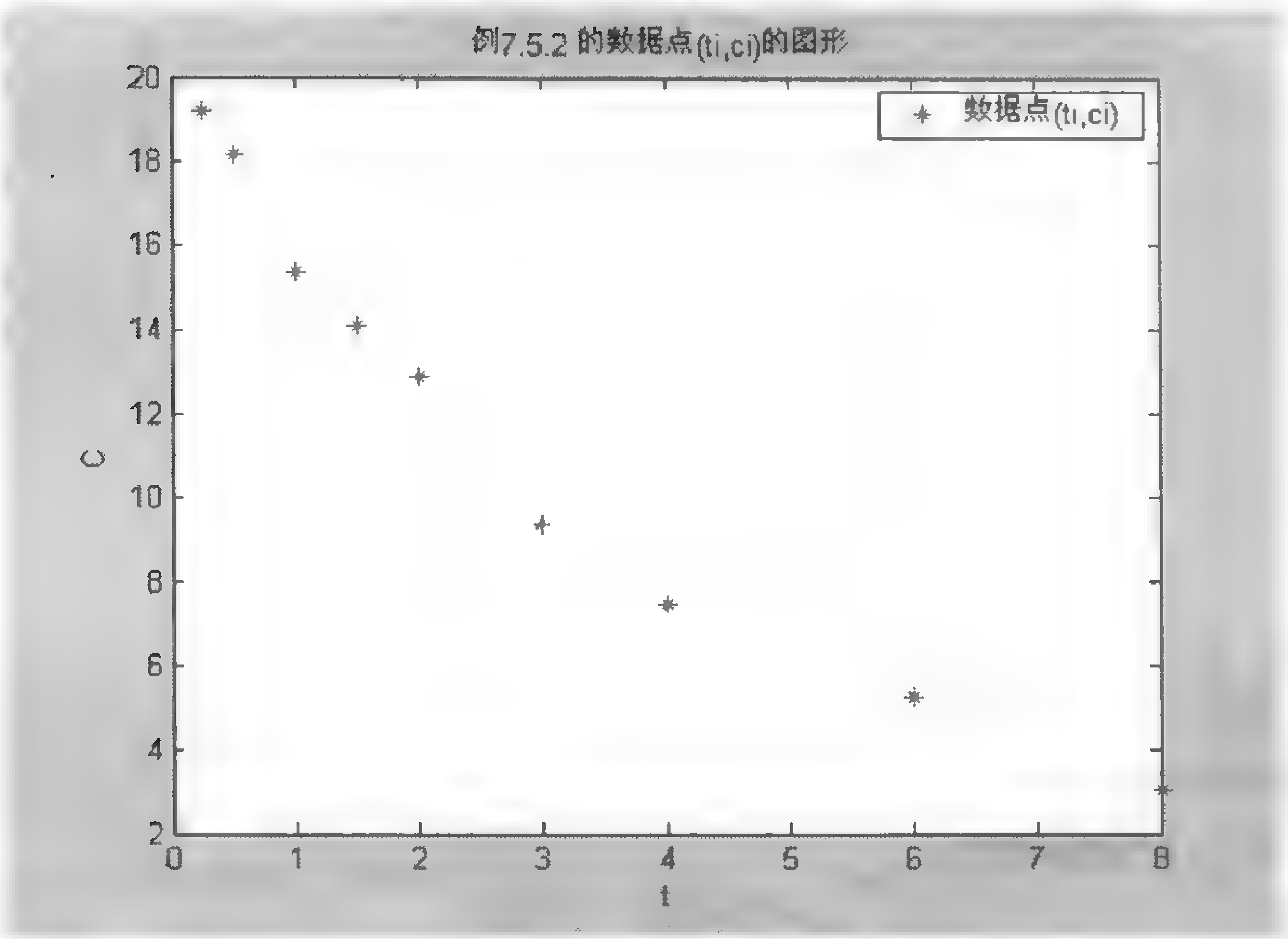
即血药浓度 $c(t)$ 按指数规律下降.

(3) 给药方案设计.

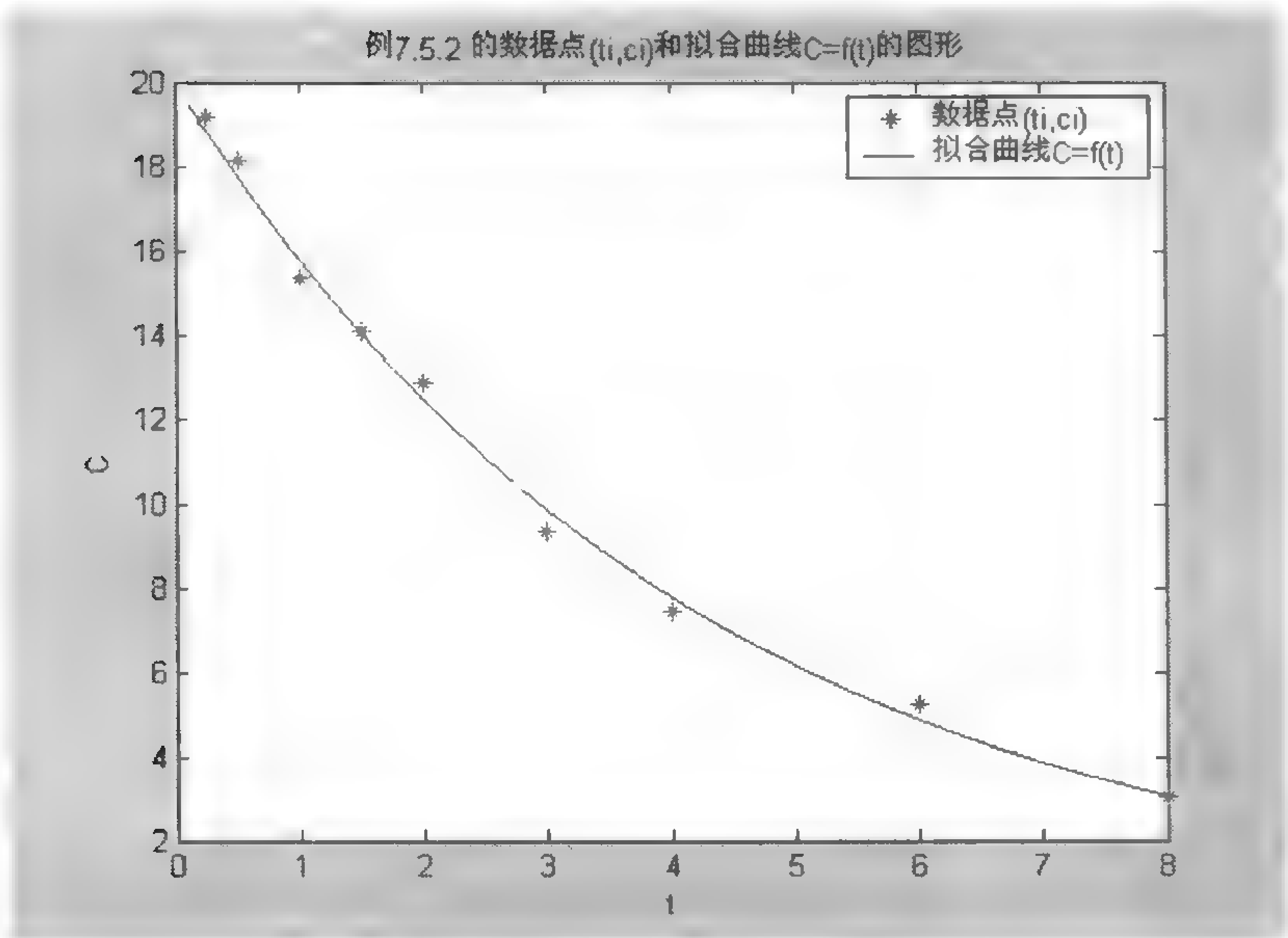
简单实用的给药方案是,每隔一定时间 τ,重复注入固定剂量 D,使血药浓度 $c(t)$ 呈周期性变化,并保持在 c_1 与 c_2 之间,如图 7-12 所示. 为此初次剂量需加大到 D_0,由式(7.21)容易得到

$$D_0=Vc_2,\quad D=V(c_2-c_1),\quad \tau=\frac{1}{k}\ln\frac{c_2}{c_1} \tag{7.23}$$

显然,当 c_1,c_2 给定后,要确定给药方案 $\{D_0,D,\tau\}$,必须知道参数 V 和 k.



(a) 血药浓度 $c(t)$ 数据图



(b) 散点图和拟合曲线

图 7-11

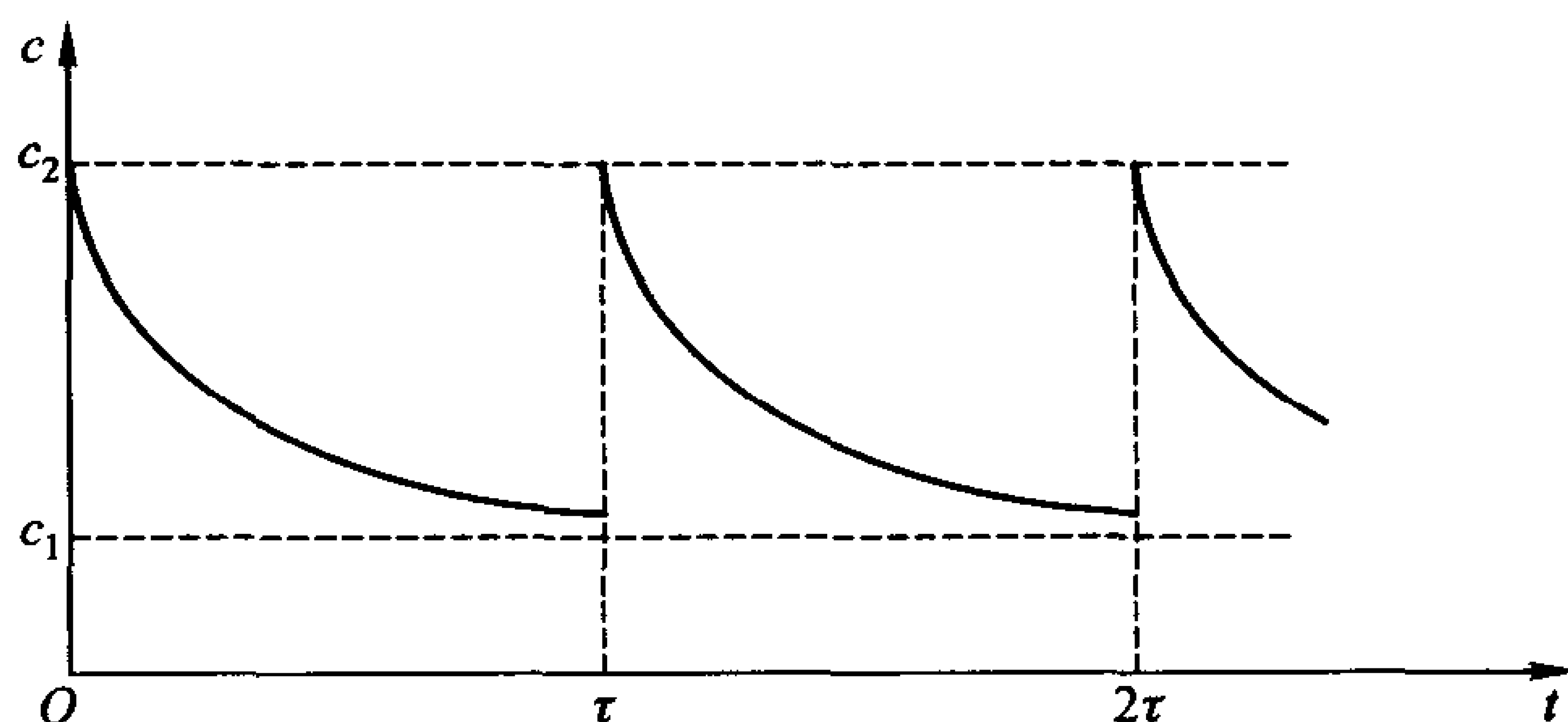


图 7-12 简单实用的给药方案中血药浓度 $c(t)$ 的图形

(4) 由实验数据作曲线拟合以确定参数.

根据图 7-11(a)血药浓度实测数据的图形,直观地看出近似于指数下降曲线,与理论分析得到的(7.21)式相符.为了用线性最小二乘法拟合(7.21)式的系数 V 和 k,先取对数得

$$\ln c=\ln(d/V)-kt. \tag{7.24}$$

记

$$y=\ln c,\quad a=-k,\quad b=\ln(d/V), \tag{7.25}$$

问题化为由数据 $t_i,y_i(i=1,2,\cdots,8)$ 拟合直线

$$y=at+b. \tag{7.26}$$

用 MATLAB 作线性最小二乘法拟合,输入程序

```
>> t =[0.25  0.5  1  1.5  2  3  4  6  8];
c =[19.21  18.15  15.36  14.10  12.89  9.32  7.45  5.24  3.01];
y = log(c);a = polyfit(t,y,1);a1 = a(1),a2 = a(2),d = 300;
T = 0.1:0.001:8;k = -a1,v = d * exp( -a2) ,f = d * exp( -k.* t)/v;
F = d * exp( -k.* T)/v; plot(t,c,'r *',T,F,'b -')
xlabel('t'), ylabel('C')
legend('数据点(ti,ci)','拟合曲线 C = f(t)')
title('例 7.5.2 的数据点(ti,ci)和拟合曲线 C = f(t)的图形')
n = length(t);fy = abs(f - c);fy2 - fy.^2;Ew = max(fy),
E1 = sum(fy)/n,E2 = sqrt((sum(fy2))/n)
```

运行后得到 $a=-0.2347$,$b=2.9943$. 由实验数据 $d=300$ mg 算出 $k=0.2347$ (1/h),$V=15.0219$ L. 数据 (t_i,r_i) 与拟合函数 $C=300\exp(-0.2347t)/15.0219$ 的最大误差 $E_w=0.5566$,平均误差 $E_1=0.3305$ 和均方根误差 $E_2=0.3673$. 画出散点图和拟合曲线,见图 7-11(b). 简单实用的给药方案中血药浓度 $c(t)$ 的图形见图 7-12.

(5) 结论.

将 k,V 和给出的 $c_1=10$ μg/ mL，$c_2=25$ μg/ mL 代入(7.24)式，得 $D_0=375.5$，$D=225.3$，$\tau=3.9$. 给药方案不妨定为 $D_0=375$ mg，$D=225$ mg，$r=4$ h.

(6) 几点说明.

① 对(7.22)式取对数后化为线性最小二乘拟合，问题固然简单了，但是拟合准则已经不是数据点(t_i,c_i)与曲线 $c(t)$ 的距离平方和最小了. 直接用(7.22)式拟合的非线性最小二乘方法将在以后介绍.

② k,V 是根据某一人的实验数据拟合得到的，对不同人做实验，结果会有不同. 特别，体重相差大的人，血液容积 V 会有较大差异. 所以实际上应对相当数量的人群进行测试、计算，也有必要对不同体重加以区分(如成人与儿童).

③ 给药方式还有静脉滴注、肌肉注射、口服等，在这些方式下微分方程(7.20)右端应有自由项. 另外，这个题目用的是一室模型，也可以构造二室模型(血液丰富的心、肺、肾等组成的中心室，和血液相对贫乏的肌肉组织构成的周边室)，或多室模型. 其他给药方式、二室模型的建立及参数估计.



习 题 7.5

1. 电压 $V=10$ V 的电池给电容器充电，电容器上 t 时刻的电压为 $v(t)=V-(V-v_0)e^{-t/\tau}$，其中 v_0 是电容器的初始电压，τ 是充电常数. 试由下面一组 t,v 数据确定 v_0 和 τ.

t(s)	0.5	1	2	3	4	5	7	9
V(V)	6.36	6.48	7.26	8.22	8.66	8.99	9.43	9.63

2. 已知数据点(t,c)列入下表，求 t 和 c 的变化规律，估计误差.

t	0.25	0.5	1	1.5	2	3	4	6	8
c	19.21	18.15	15.36	14.10	12.89	9.32	7.45	5.24	3.01

3. 给出一组实验数据点(x_i,y_i)的横坐标向量为 $\boldsymbol{x}=(-6.8\quad -5.10\quad -4.5\quad -3.6\quad -3.4\quad -2.6\quad -2.5\quad -2.1\quad -1.5\quad -2.7\quad -3.6)$，纵横坐标向量为 $\boldsymbol{y}=(165.60\quad 59.17\quad 41.66\quad 25.92\quad 22.37\quad 13.47\quad 12.87\quad 11.87\quad 6.69\quad 14.87\quad 24.22)$，试用线性变换和线性最小二乘法求拟合曲线，并用(7.2)，(7.3)和(7.4)式估计其误差，作出拟合曲线.

7.6 函数逼近及其 MATLAB 程序

本节讨论与数据拟合相近的所谓函数逼近. 前面讲的曲线拟合是已知一组离散数据(x_i,y_i)，$i=1,2,\cdots,n$，选择一个较简单的函数 $f(x)$（如多项式），使在

一定准则(如最小二乘准则)下,最接近这组数据.如果已知一个较为复杂的连续函数 $y(x)$,$x\in[a,b]$,要求选择一个较简单的函数 $f(x)$,在一定准则下最接近 $y(x)$,就是所谓**函数逼近**.与曲线拟合的最小二乘准则相应,函数逼近常用的一种准则是最小平方逼近,即

$$J=\int_a^b[f(x)-y(x)]^2\mathrm{d}x \tag{7.27}$$

达到最小.与曲线拟合一样,选一组函数 $\{r_k(x),k=1,2,\cdots,m\}$ 构造 $f(x)$,即令

$$f(x)=a_1r_1(x)+\cdots+a_mr_m(x), \tag{7.28}$$

代入(7.27)式,求 $a_1,a_2,\cdots,a_m$ 使 J 达到极小.利用极值必要条件可得

$$(\boldsymbol{RR})\boldsymbol{A}=(\boldsymbol{RY}), \tag{7.29}$$

其中

$$(\boldsymbol{RR})=\begin{pmatrix}\int_a^b r_1^2(x)\mathrm{d}x & \cdots & \int_a^b r_1(x)r_m(x)\mathrm{d}x\\ \vdots & & \vdots\\ \int_a^b r_m(x)r_1(x)\mathrm{d}x & \cdots & \int_a^b r_m^2(x)\mathrm{d}x\end{pmatrix},$$

$$\boldsymbol{A}=(a_1,a_2,\cdots,a_m)^{\mathrm{T}},\ (\boldsymbol{RY})=\left(\int_a^b y(x)r_1(x)\mathrm{d}x,\cdots\cdots,\int_a^b y(x)r_m(x)\mathrm{d}x\right)^{\mathrm{T}},$$

当$(\boldsymbol{RR})$可逆时,$\boldsymbol{A}$ 有唯一解.

最简单的当然是用多项式函数逼近,即选 $r_1(x)=1,r_2(x)=x,r_3(x)=x^2,\cdots$. 并且如果能使 $\int_a^b r_i(x)r_j(x)\mathrm{d}x=0\ (i\neq j)$,$(\boldsymbol{RR})$将是对角矩阵,计算也将大大简化.满足这种性质的多项式称正交多项式.

勒让德(Legendre)多项式是在$[-1,1]$区间上的正交多项式,它的表达式为

$$P_0(x)=1,\quad P_k(x)=\frac{1}{2^k k!}\cdot\frac{\mathrm{d}^k}{\mathrm{d}x^k}(x^2-1)^k,\quad k=1,2,\cdots. \tag{7.30}$$

可以证明,

$$\int_{-1}^1 P_i(x)P_j(x)\mathrm{d}x=\begin{cases}0, & i\neq j,\\ \dfrac{2}{2i+1}, & i=j,\end{cases} \tag{7.31}$$

$$P_{k+1}(x)=\frac{2k+1}{k+1}xP_k(x)-\frac{k}{k+1}P_{k-1}(x),\quad k=1,2,\cdots.$$

最佳均方逼近是利用一系列已知的函数来逼近未知的函数,只要知道参加所有的逼近的已知函数即可.现提供最佳均方逼近的主程序如下.

最佳均方逼近的 MATLAB 主程序

输入量:f 是参加所有的逼近的已知函数名组成的字符串矩阵,每个函数名占一行,每行的字符个数要相同,不同时用空格补齐;x 和 y 是参加逼近的原始数据的第一个坐标和函数值,它们的长度要相同;xx 是要进行插值点的第一坐标组成的向量.

输出量:a 是参加所有的逼近函数的系数组成的向量,a 的长度与矩阵 f 列的长度相同;WE 为均方误差;yy_1 是与 xx 对应的插值点的函数值.

```
function [yy1,a,WE] = zjjfbj(f,X,Y,xx)
m = size(f);n = length(X);m = m(1);b = zeros(m,m); c = zeros(m,1);
if n ~ = length(Y)
     error('X 和 Y 的维数应该相同')
end
for j =1:m
     for k =1:m
          b(j,k) =0;
          for i =1:n
           b(j,k) =b(j,k) + feval(f(j,:),X(i)) * feval(f(k,:),X(i));
          end
     end
     c(j) =0;
     for i =1:n
          c(j) = c(j) + feval(f(j,:),X(i)) * Y(i);
     end
end
a = b\c;% 产生系数向量 a
WE = 0;
for i =1:n
       ff =0;
    for j =1:m
      ff = ff + a(j) * feval(f(j,:),X(i));
    end
      WE = WE + (Y(i) - ff) * (Y(i) - ff);
end
if nargin = =3
     return;
end
```

```
yy = [];
for i = 1:m
    l = [];
    for j = 1:length(xx)
        l = [l,feval(f(i,:),xx(j))];
    end
    yy = [yy l'];
end
yy = yy * a; yy1 = yy'; a = a';WE;
```

例 7.6.1 对数据 X 和 Y,用函数 $y=1,y=x,y=x^2$ 进行逼近,计算 $x=6.5$ 处的函数值,并估计误差.其中 $X=(1\quad 3\quad 4\quad 5\quad 6\quad 7\quad 8\quad 9)$;$Y=(-11\quad -13\quad -11\quad -7\quad -1\quad 7\quad 17\quad 29)$.

解 (1) 建立函数 M 文件

```
function f0 = fun0(X)
f0 = 1;
function f1 = fun1(X)
f1 = X;
function f2 = fun2(X)
f2 = X ^2;
```

(2) 在 MATLAB 工作窗口输入程序

```
>> X = [1  3  4  5  6  7  8  9];
Y = [ -11  -13  -11  -7  -1  7  17  29];
f = ['fun0';'fun1';'fun2']; [yy,a,WE] = zjjfbj(f,X,Y,6.5)
```

运行后屏幕显示参加所有的逼近函数的系数组成的向量 a,均方误差 WE,插值点 xx 的函数值 yy 如下

```
yy =
  2.75000000000003
a =
  -7.00000000000010  -4.99999999999995  1.00000000000000
WE =
  7.172323350269439e-027
```

即,逼近函数

$$f(x)=-7.000\,000\,000\,000\,10-4.999\,999\,999\,999\,95\,x+1.000\,000\,000\,000\,00\,x^2,$$

在 $x=6.5$ 处的函数值为 $y=2.750\,000\,000\,000\,03$,误差为

$$WE=7.172\,323\,350\,269\,44\times10^{-27}.$$

例 7.6.2 对数据 X 和 Y,用函数 $y=1,y=x,y=x^2,y=\cos x,y=e^x,y=\sin x$ 进行逼近,其中 $X=(0\ 0.50\ 1.00\ 1.50\ 2.00\ 2.50\ 3.00)$,$Y=(0\ 0.4794$

0.8415 0.9815 0.9126 0.5985 0.1645).

解 (1) 建立函数 M 文件

```
function f0 = fun0(X)
f0 =1;
function f1 = fun1(X)
    f1 = X;
function f2 = fun2(X)
f2 = X ^2;
function f3 = fun3(X)
    f3 = exp(X);
function f4 = fun4(X)
  f4 = cos(X);
function f5 = fun5(X)
    f5 = sin(X);
```

(2) 在 MATLAB 工作窗口输入程序

```
>>  X = [ 0  0.50  1.00  1.50  2.00  2.50  3.00];
Y = [0  0.4794  0.8415  0.9815  0.9126  0.5985  0.1645];
f = ['fun0';'fun1';'fun2';'fun3';'fun4';'fun5'];xx = 0:0.2:3;
[yy,a,WE] = zjjfbj(f,X,Y, xx), plot(X,Y,'ro',xx,yy,'b-')
```

运行后屏幕显示参加所有的逼近函数的系数组成的向量 a,均方误差 WE,插值点 xx 的函数值 yy 和图 7－13 如下.

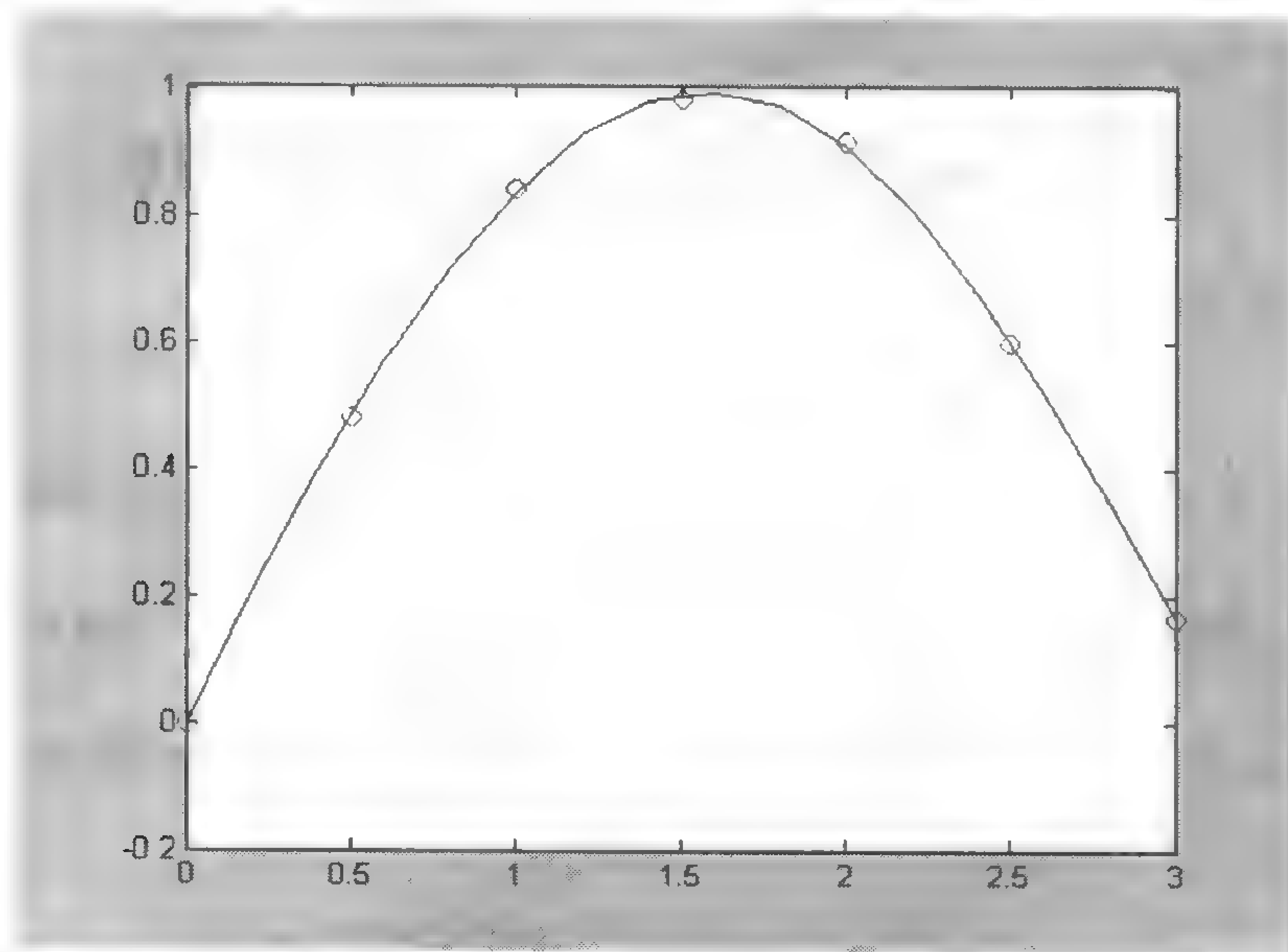


图 7－13 最佳逼近函数的图形

```
yy =   Columns 1 through 7
-0.0005   0.2037   0.3939   0.5656   0.7141   0.8348   0.9236
  Columns 8 through 14
0.9771   0.9926   0.9691   0.9069   0.8080   0.6766   0.5191
```

```
  Columns 15 through 16
    0.3444     0.1642
 a =0.3828    0.4070    -0.3901    0.0765    -0.4598    0.5653
 WE  = 1.5769e-004
```

即,最佳逼近函数为

```
y =0.3828 +0.4070 * x -0.3901 * x^2 +0.07658 * exp(x)  -0.4598 * cos(x)  +
0.5653 * sin(x).
```



习 题 7.6

1. 对数据 X 和 Y,用函数 $y=1,y=x,y=x^2,y=\tan x,y=e^x,y=\sin x$ 进行逼近,其中 $X=(0\quad 0.40\quad 1.00\quad 1.50\quad 2.00\quad 2.50\quad 3.00)$,$Y=(0\quad 0.3794\quad 0.8415\quad 0.9815\quad 0.9126\quad 0.5985\quad 0.1645)$.

2. 对数据 X 和 Y,用函数 $y=1,y=x,y=x^2$ 进行逼近,其中

$X=(1\quad 2\quad 3\quad 4\quad 5\quad 6\quad 7\quad 8)$,$Y=(-11\quad -13\quad -11\quad -7\quad -1\quad 8\quad 17\quad 29)$.

7.7 三角多项式逼近及其 MATLAB 程序

在高等数学中我们学过三角级数

$$\frac{a_0}{2}+\sum_{n=1}^{\infty}(a_n\cos nx+b_n\sin nx),\tag{7.32}$$

其中 $a_0,a_n,b_n(n=1,2,3,\cdots)$ 都是常数. 三角级数(7.32)是由三角函数系

$$1,\cos x,\sin x,\cos 2x,\sin 2x,\cdots,\cos nx,\sin nx,\cdots\tag{7.33}$$

构成. 三角函数系(7.33)在区间 $[-\pi,\pi]$ 上正交. 经常使用的三角级数是傅里叶级数.

定义 7.1(傅里叶级数) 设 $f(x)$ 是以 2π 为周期的周期函数,且在区间 $[-\pi,\pi]$ 上分段连续,则 $f(x)$ 的傅里叶级数

$$S(x)=\frac{a_0}{2}+\sum_{n=1}^{\infty}(a_n\cos nx+b_n\sin nx),\tag{7.34}$$

其中傅里叶系数为

$$a_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\cos nx\mathrm{d}x,\qquad n=0,1,2,\cdots,\tag{7.35}$$

$$b_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\sin nx\mathrm{d}x,\qquad n=1,2,\cdots.\tag{7.36}$$

傅里叶级数的收敛性由下面定理给出.

定理 7.1(傅里叶级数收敛定理) 设以 2π 为周期的周期函数 $f(x)$ 在区间 $[-\pi,\pi]$ 上连续或只有有限个第一类间断点,且至多只有有限个极值点,则 $f(x)$ 的傅里叶级数收敛,且

$$\frac{a_0}{2}+\sum_{n=1}^{\infty}(a_n\cos nx+b_n\sin nx)=\begin{cases}f(x), & \text{当 } x \text{ 是 } f(x) \text{ 的连续点时},\\ \frac{1}{2}[f(x-0)+f(x+0)], & \text{当 } x \text{ 是 } f(x) \text{ 的间断点时},\end{cases} \tag{7.37}$$

其中 $f(x-0)$ 和 $f(x+0)$ 分别表示 $f(x)$ 在 x 处的左极限和右极限.

定义 7.2(三角多项式) 具有如下形式的级数

$$T_m(x)=\frac{a_0}{2}+\sum_{k=1}^{m}(a_k\cos kx+b_k\sin kx) \tag{7.38}$$

称为 m **阶三角多项式**.

定理 7.2(离散傅里叶级数定理) 设 $f(x)$ 是以 2π 为周期的周期函数,在区间 $[-\pi,\pi]$ 上的 n 等分点和区间端点 $x_i=-\pi+\frac{2i\pi}{n}\ (i=0,1,2,\cdots,n)$ 处 $y_i=f(x_i)$,且 $2m<n$,则存在(7.38)式的三角多项式 $T_m(x)$,使得均方误差

$$R_m(x)=\frac{1}{n}\sum_{j=1}^{n}[f(x_j)-T_m(x_j)]^2 \tag{7.39}$$

的值最小,其中三角多项式 $T_m(x_j)$ 的系数计算公式为

$$a_j=\frac{2}{n}\sum_{k=1}^{n}f(x_k)\cos jx_k \quad (j=0,1,2,\cdots,m), \tag{7.40}$$

$$b_j=\frac{2}{n}\sum_{k=1}^{n}f(x_k)\sin jx_k \quad (j=1,2,\cdots,m). \tag{7.41}$$

离散傅里叶级数定理给出了对数据点集 $\{(x_i,y_i)\}_{i=0}^{n}$ 进行曲线拟合的三角多项式. 数据点的个数 n 越大,三角多项式 $T_m(x_j)$ 的系数越接近傅里叶级数的系数.

根据三角多项式(7.38)及其系数计算公式(7.40)、(7.41)和均方误差(7.39),现编写下面的计算三角多项式的 MATLAB 程序,并保存名为 `sanjiao.m`的 M 文件. 我们可以利用此程序计算 m 阶三角多项式 $T_m(x)$ 的系数 $a_j(j=0,1,2,\cdots,m)$ 和 $b_j(j=1,2,\cdots,m)$ 构成的系数矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$,$T_m(x)$ 在 $\boldsymbol{X}_1$ 处的值 $\boldsymbol{Y}_1$,用(7.39)式计算的 $f(x)$ 与 $T_m(x)$ 均方误差 R_m.

计算三角多项式的 MATLAB 主程序

输入的量：数据点的向量(X,Y)，其中向量 X 的元素是 $n+1$ 个等距离的点 $x_i=-\pi+\frac{2i\pi}{n}$ $(i=0,1,2,\cdots,n)$，向量 Y 的元素是 $y_i=f(x_i)$，X_1是数或向量，m 是三角多项式 $T_m(x)$的阶，且 $2m<n$.

输出的量：矩阵 A 和 B 分别是由 m 阶三角多项式 $T_m(x)$的系数 $a_j(j=0,1,2,\cdots,m)$和 $b_j(j=1,2,\cdots,m)$构成的系数矩阵，Y_1是 $T_m(x)$在 X_1处的值，R_m是用(7.39)式计算的$f(x)$与 $T_m(x)$均方误差.

```
function [A,B,Y1,Rm] = sanjiao(X,Y,X1,m)
n = length(X) -1;max1 = fix((n -1)/2);
if m >max1
     m = max1;
end
A = zeros(1,m +1);B - zeros(1,m +1);
Ym = (Y(1) +Y(n +1))/2;Y(1) = Ym;
Y(n +1) = Ym;A(1) =2 * sum(Y)/n;
for i =1:m
      B(i +1) = sin(i * X) * Y';
      A(i +1) = cos(i * X) * Y';
end
  A =2 * A/n;B =2 * B/n;
  A(1) =A(1)/2;Y1 =A(1);
for k =1:m
      Y1 =Y1 +A(k +1) * cos(k * X1) + B(k +1) * sin(k * X1);
      Tm =A(1) +A(k +1).* cos(k * X) + B(k +1).* sin(k * X);k =k +1;
end
Y;Tm; Rm =(sum(Y - Tm).^2)/n;
```

例 7.7.1 根据$[-\pi,\pi]$上的 $n=13,60,350$ 个等距横坐标点 $x_i=-\pi+\frac{2i\pi}{n}$ $(i=0,1,2,\cdots,n)$和函数$f(x)=2\sin\frac{x}{3}$.

(1) 求$f(x)$的 6 阶三角多项式逼近，计算均方误差；

(2) 将这三个三角多项式分别与 $f(x)$ 的傅里叶级数 $f(x)=\frac{18\sqrt{3}}{\pi}\sum_{n=1}^{\infty}(-1)^{n+1}\frac{n}{9n^2-1}\sin nx$ 的前 6 项进行比较；

(3) 利用三角多项式分别计算 $X_i=-2,2.5$ 的值；

(4) 在同一坐标系中，画出函数$f(x)$，$n=13,60,350$ 的三角多项式和数据点的图形.

解 (1) 输入程序

```
>> X1 = -pi:2*pi/13:pi;
Y1 =2*sin(X1/3);X1i =[-2,2.5];
[A1,B1,Y11,Rm1] = sanjiao(X1,Y1,X1i,6),
  X2 = -pi:2*pi/60:pi;Y2 =2*sin(X2/3);
[A2,B2,Y12,Rm2] = sanjiao(X2,Y2,X1i,6)
X3 = -pi:2*pi/350:pi;Y3 =2*sin(X3/3);
[A3,B3,Y13,Rm3] = sanjiao(X3,Y3,X1i,6)
X1i =[-2,2.5];Y1 =2*sin(X1i/3)
for n=1:6
bi=(-1)^(n+1)*18*sqrt(3)*n/(pi*(9*n^2-1)).
end
```

运行后输出$f(x)$的 $n=13,60,350$ 时，三个 6 阶三角多项式的系数矩阵 $\boldsymbol{A}_1,\boldsymbol{A}_2,\boldsymbol{A}_3$，均方误差 R_{1m},R_{2m},R_{3m}. $f(x)$的傅里叶级数(7.34) $Tm1,Tm2,Tm3$ 及其前 6 项的系数 $A1,B1,A2,B2,A3,B3$，利用三个 6 阶三角多项式分别计算 $X_i=-2,2.5$的值 Y_{1i},Y_{2i},Y_{3i}，给定的$f(x)$的傅里叶级数的前 6 项的系数 b_i，将这些值整理后列入表 7-7 中.

当 $n=13$ 时的 6 阶三角多项式为

$$T_5(x)=1.2189\sin x-0.5234\sin 2x+0.3053\sin 3x-0.1857\sin 4x+0.1018\sin 5x-0.0326\sin 6x.$$

当插入点增至 60 个和 350 个时，6 阶三角多项式的系数变化很小. 当数据点的个数增加得越多，则三角多项式的前 6 项的系数越接近$f(x)$的傅里叶级数的前 6 项的系数. 但是，利用三个 6 阶三角多项式分别计算 $X_i=-2,2.5$ 的值 Y_{1i},Y_{2i},Y_{3i}与对应的函数$f(x)$的值的误差增大. 具体数值如表 7-7 所示.

表 7-7 例 7.7.1 的计算结果

	三角多项式的系数			傅里叶级数的系数
	$n=13$	$n=60$	$n=350$	
b_1	1.218 9	1.239 5	1.240 5	1.240 5
b_2	-0.523 4	-0.565 1	-0.567 0	-0.567 1
b_3	0.305 3	0.369 1	0.372 1	0.372 1
b_4	-0.185 7	-0.273 5	-0.277 5	-0.277 6
b_5	0.101 8	0.216 5	0.221 4	0.221 5
b_6	-0.032 6	-0.178 3	-0.184 2	-0.184 3
R_{im}	9.940 6e-032	2.569 0e-031	4.578 2e-033	Y1 =2sin(X1i/3)值
Y_{ii}	-1.197 5 1.590 8	-1.158 8 1.648 4	-1.157 0 1.651 6	-1.236 7 1.480 4

(2)画图,输入程序

```
>>X1 = -pi:2*pi/13:pi;Y1 =2*sin(X1/3);
Xi = -pi:0.001:pi; f =2*sin(Xi/3);
[A1,B1,Y1i,R1m] = sanjiao(X1,Y1,Xi,6);X2 = -pi:2*pi/60:pi;
Y2 =2*sin(X2/3); X3 = -pi:2*pi/350:pi;Y3 =2*sin(X3/3);
[A2,B2,Y2i,R2m] = sanjiao(X2,Y2,Xi,6);
[A3,B3,Y3i,R3m] = sanjiao(X3,Y3,Xi,6);
plot(X1,Y1,'r*', Xi, Y1i,'b-',Xi, Y2i,'g--', Xi, Y3i, 'm:', Xi, f,
'k-.')
xlabel('x'),ylabel('y')
legend('数据点(xi,yi)','n=13 的三角多项式','n=60 的三角多项式','n=
350 的三角多项式','函数 f(x)')
title('例 7.7.1 的数据点(xi,yi)、n=13,60,350 的三角多项式 T3 和函数
f(x)的图形')
```

运行后,在同一坐标系中,画出函数 $f(x)$,$n=13,60,350$ 的三角多项式和数据点的图形,见图 7-14.

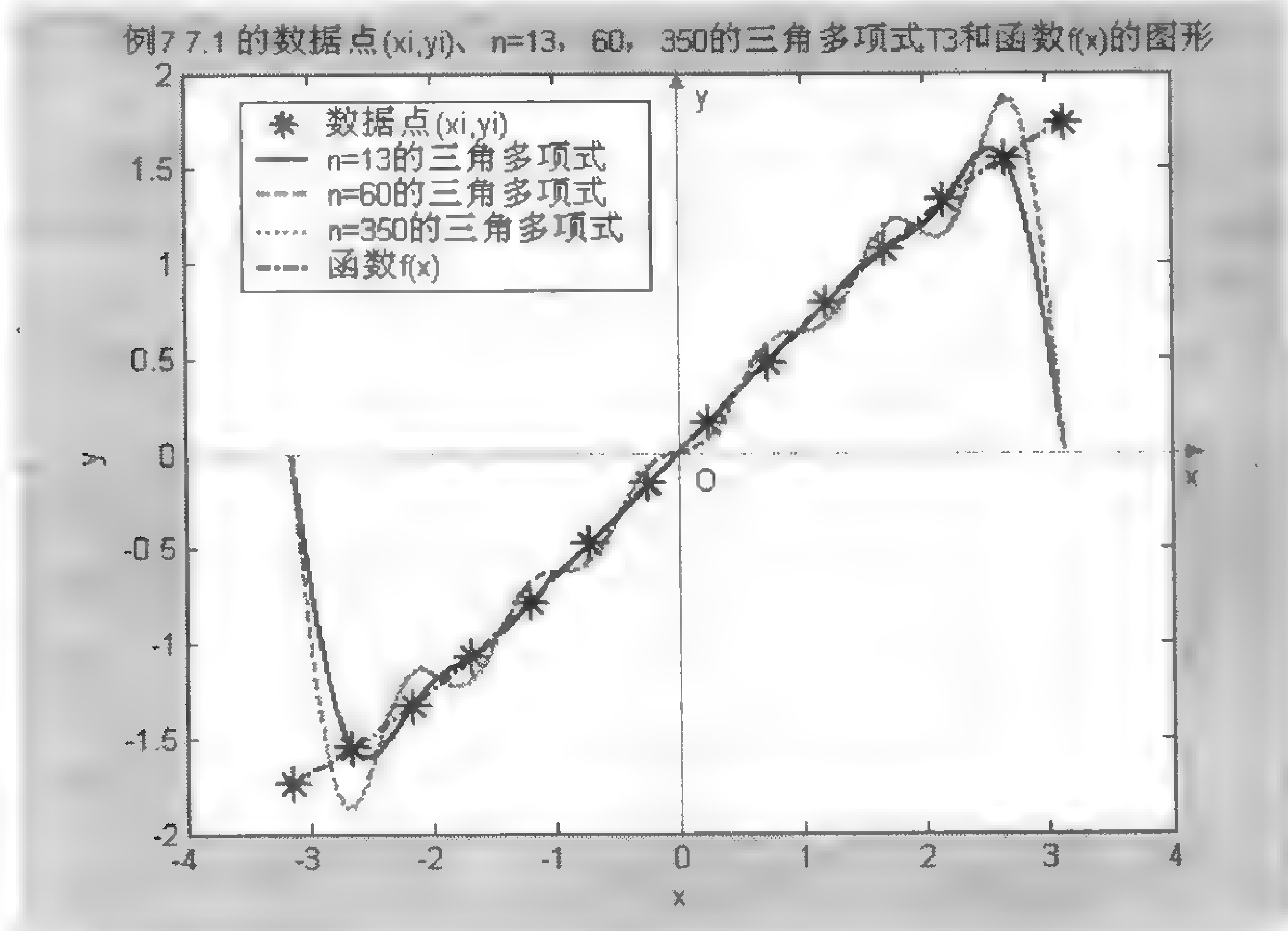
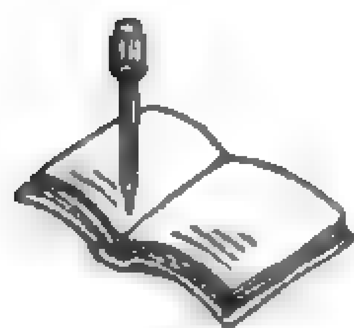


图 7-14 例 7.7.1 的函数 $f(x)$,$n=13,60,350$ 的三角多项式和数据点的图形



习题 7.7

1. 根据$[-\pi,\pi]$上的$n=13,60,350$个等距横坐标点$x_i=-\pi+\frac{2i\pi}{n}$ $(i=0,1,2,\cdots,n)$和下列函数.

① $f(x)=3\sin\frac{x}{5}$；　　② $f(x)=3\cos 2x$；

③ $f(x)=\begin{cases}3x^2+1, & -\pi\leqslant x\leqslant 0,\\ e^{2x}, & 0<x\leqslant\pi;\end{cases}$　　④ $f(x)=|x|$.

(1) 求$f(x)$的6阶三角多项式逼近，计算均方误差；

(2) 将这三个三角多项式分别与$f(x)$的傅里叶级数$f(x)=\frac{18\sqrt{3}}{\pi}\sum_{n=1}^{\infty}(-1)^{n+1}\frac{n}{9n^2-1}\sin nx$的前6项进行比较；

(3) 利用三角多项式分别计算$X_i=-2,2.5$的值；

(4) 在同一坐标系中，画出函数$f(x)$，$n=13,60,350$的三角多项式和数据点的图形.

2. 在洛杉矶郊区11月8日的温度(华氏温度)如下表所示，采用24小时制.

时间	1	2	3	4	5	6	7	8	9	10	11	12
温度	58	58	58	58	57	57	57	58	60	64	67	68
时间	13	14	15	16	17	18	19	20	21	22	23	24
温度	66	66	65	64	63	63	62	61	60	60	59	58

(1) 求三角多项式T_8；

(2) 在同一坐标系中，画出T_8的三角多项式和24个数据点的图形；

(3) 使用本地的温度情况，重新求解问题(1)和(2).

3. 在化工生产中常常需要知道丙烷在各种温度T和压力p下的导热系数K. 下面是实验得到的一组数据：

T(℃)	p(10^3kN/ m^2)	K	T(℃)	p(10^3kN/ m^2)	K
68	9.798	0.085	106	9.792	0.070
68	13.324	0.090	106	14.277	0.075
87	9.008	0.076	140	9.656	0.061
87	13.355	0.081	140	12.463	0.065

试求$T=99$ ℃和$p=10.3\times10^3$ kN/ m^{2}下的K.

7.8 随机数据点上的二元拟合及其 MATLAB 程序

随机数据点是对单调数据点而言的，这里的随机数据点$\{(x_i,y_i,z_i)\}_{i=0}^{n}$是

指数列$\{x_i\}_{i=0}^{n}$和$\{y_i\}_{i=0}^{n}$不必是单调序列(例如,随机点列),这样曲面上的点(x_i,y_i,z_i) $(i=1,2,\cdots,n)$是散乱的、无序的.所谓拟合是在区域$D=[x_1,x_m]\times[y_1,y_n]$上,或者在$D$的分片子域$D_{ij}$上,构造与被插值函数$z=f(x,y)$的曲面上的随机数据点$(x_i,y_i,z_i)$ $(i=1,2,\cdots,n)$最为接近的较简单函数$P(x,y)$,拟合曲面$P(x,y)$不必要过所有的节点(x_i,y_i,z_i) $(i=1,2,\cdots,n)$,使得简单函数$P(x,y)$最佳逼近被插值函数$f(x,y)$ (见下面的图形),这也是随机数据点上的二元拟合的主要任务.在MATLAB系统中,为我们提供了计算随机数据点上的二元拟合的程序 griddata.m,计算时直接调用.程序 griddata.m 主要用于将随机数据点上的数据变换网格坐标拟合曲面.程序 griddata 的功能是计算常用的二元拟合方法中最近邻内插法(Nearest neighbor interpolation)、三角基线性内插法(Triangle - based linear interpolation)(缺省值)、三角基三次内插法(Triangle - based cubic interpolation)和 MATLAB 4 网格化坐标方法(MATLAB 4 griddata method).计算时输入的随机数据点$(\boldsymbol{x},\boldsymbol{y})$中的向量$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$,$\boldsymbol{y}=(y_1,y_2,\cdots,y_n)$的元素通常不必是单调排列,用网格化命令[X,Y] = meshgrid(x,y)将$\boldsymbol{x}$,$\boldsymbol{y}$化为二元网格坐标X,Y,然后调用 griddata 程序计算.该程序有几种调用格式如表7-8所示.

表7-8 随机数据点上的二元拟合的MATLAB命令

随机数据点上的二元拟合的 MATLAB 命令 griddata	功　　能
Z1 = griddata(X,Y,Z,X1,Y1)	Z1 = griddata(X,Y,Z,X1,Y1)命令的主要功能是通常对于随机数据点向量(X,Y,Z)中的数据拟合函数$Z=F(X,Y)$的曲面. griddata 在由(X_i,Y_i)产生的Z_i处内插这张曲面,并且该曲面总是经过数据点.其中X_i和Y_i是由 meshgrid 命令产生的均匀网格坐标.
[X1,Y1,Z1] = griddata(X,Y,Z,X1,Y1)	[X1,Y1,Z1] = griddata(X,Y,Z,X1,Y1)除了返回Z_i以外,也返回X_i和Y_i,其中返回的X_i和Y_i是[X1,Y1] = meshgrid(X1,Y1)产生的结果.
[...] = griddata(...,'method')	[...] = griddata(...,'method')的主要功能是用户指定二元内插法的方法.用户不输入具体的方法时,按三角基线性内插法计算.'method'可用以下方法替换: 'nearest' 表示二元最近邻内插法; 'linear' 表示三角基线性内插法(缺省值); 'cubic' 表示三角基三次内插法; 'v4' 表示 MATLAB 4 网格化坐标方法.

说明:定义的拟合数据的曲面的类型中,'cubic'和'v4'方法产生光滑曲面,而'linear'和'nearest'方法产生有突变性的曲面,这些突变点分别产生在不连续点和一阶导数不连续点.除了'v4'方法以外,其余的方法都是基于一种数据的 Delaunay 三角剖分.X_i可以是行向量,此时它指定每一行均为X_i的矩阵,同样,Y_i可以是列向量,此时它指定每一列均为Y_i的矩阵.参考例 7.8.1 的图 7-15 至图 7-18.

例 7.8.1 设节点(X,Y,Z)中的X和Y分别是在区间$[-3,3]$和$[-2.5,3.5]$上的 50 个随机数,Z是函数$z=7-3x^3e^{-x^2-y^2}$在(X,Y)的值,拟合点(X_i,Y_i)中的$X_i=-3:0.2:3$,$Y_i=-2.5:0.2:3.5$.分别用二元拟合方法中最近邻内插法、三角基线性内插法、三角基三次内插法和 MATLAB 4 网格化坐标方法计算在(X_i,Y_i)处的值,作出它们的图形,并与被拟和曲面进行比较.

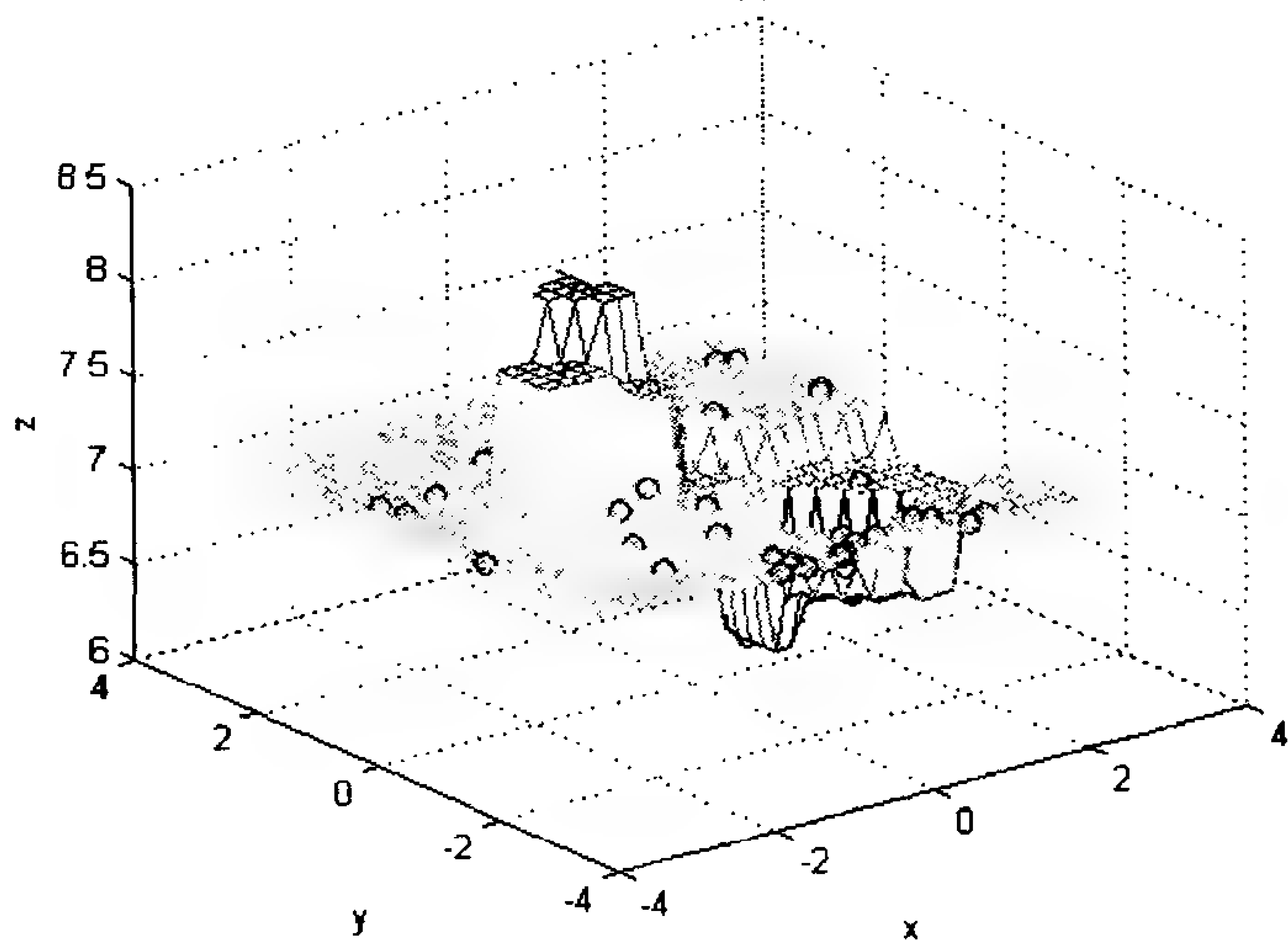
解 (1) 最近邻内插法.输入程序

```
>> x = rand(50,1);
y = rand(50,1);% 生成 50 个 1 元均匀分布随机数 x 和 y, x,y .
X = -3 + (3 - ( -3)) * x;% 利用 x 生成的随机变量.
Y = -2.5 + (3.5 - ( -2.5)) * y;% 利用 y 生成的随机变量.
Z = 7 - 3 * X.^3 .* exp( -X.^2 - Y.^2); % 在每个随机点(X,Y)处计算 Z 的
值.
X1 - -3:0.2:3;
Y1 = -2.5:0.2:3.5;
[XI,YI] = meshgrid(X1,Y1);     % 将坐标(XI,YI)网格化.
Z1 = griddata(X,Y,Z,XI,YI,'nearest')% 计算在每个插值点(XI,YI)处的
插值 ZI.
mesh(XI,YI, ZI)            % 作二元拟合图形.
xlabel('x'), ylabel('y'), zlabel('z'),
title('用最近邻内插法拟合函数 z =7 -3 x^3 exp( -x^2 -- y^2)的曲面和节
点的图形')
% legend('拟合曲面','节点(xi,yi,zi)')
hold on                    % 在当前图形上添加新图形.
plot3(X,Y,Z,'bo')          % 用蓝色小圆圈画出每个节点(X,Y,Z).
hold of                    % 结束在当前图形上添加新图形.
```

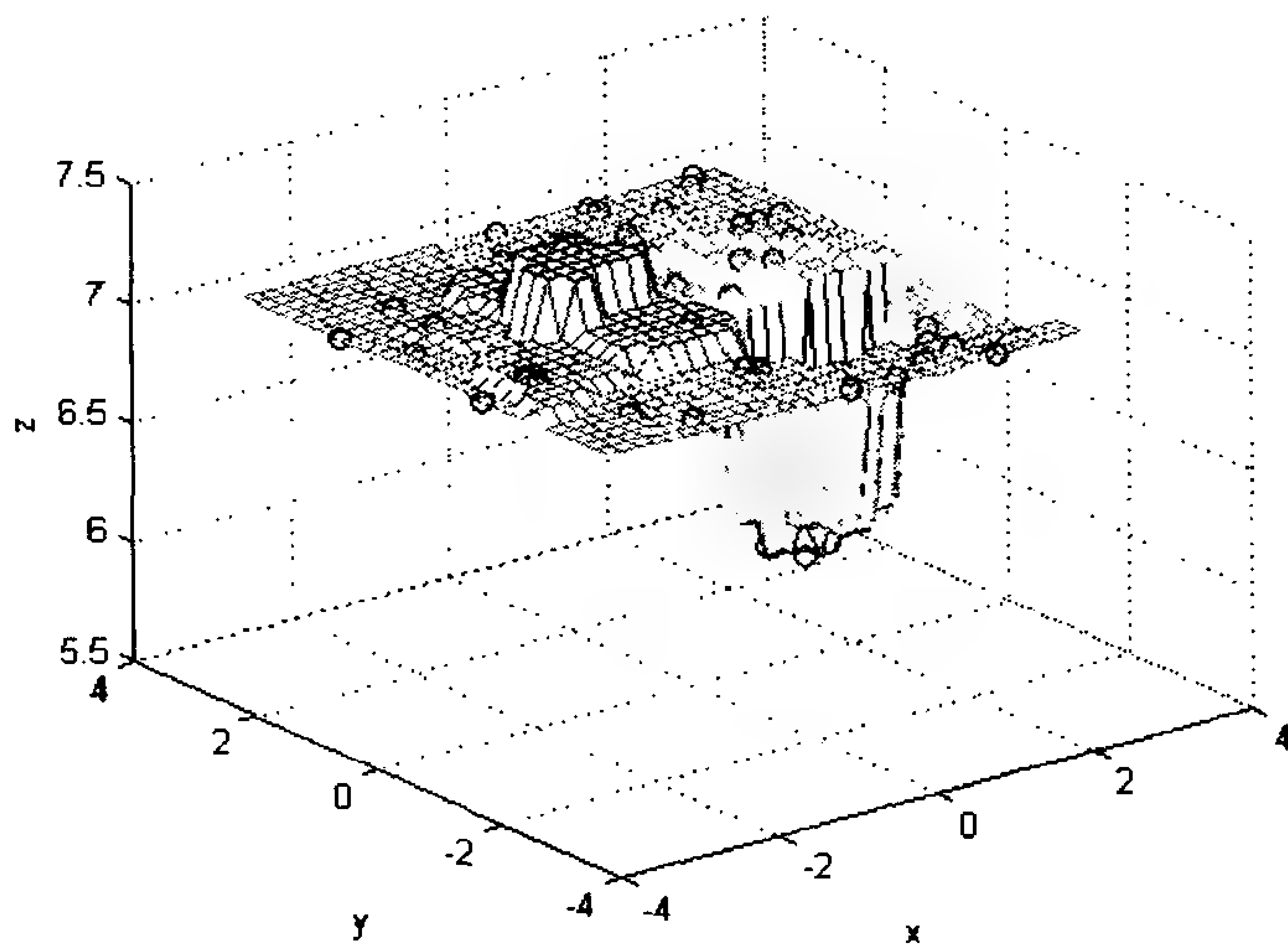
运行后屏幕显示用最近邻内插法拟合函数$z=7-3x^3e^{-x^2-y^2}$在两组不同节点处的曲面(见图 7-15)及其插值Z_i(略).由图 7-15 可见,拟合曲面的峰和谷的个数及其他的性态都随节点值的变化而变化,并且拟合曲面具有突变性,凹凸性不稳定,曲面不光滑,这些突变点分别产生在不连续点和一阶导数不连续点.

(2) 三角基线性内插法.

输入程序



(a)



(b)

图 7－15 用最近邻内插法拟合函数 $z=7-3x^3e^{-x^2-y^2}$ 的曲面和节点的图形

```
>> x = rand(50,1);
y = rand(50,1); % 生成 50 个 1 元均匀分布随机数 x 和 y, x,y
X = -3 + (3 -( -3)) * x;% 利用 x 生成的随机变量.
```

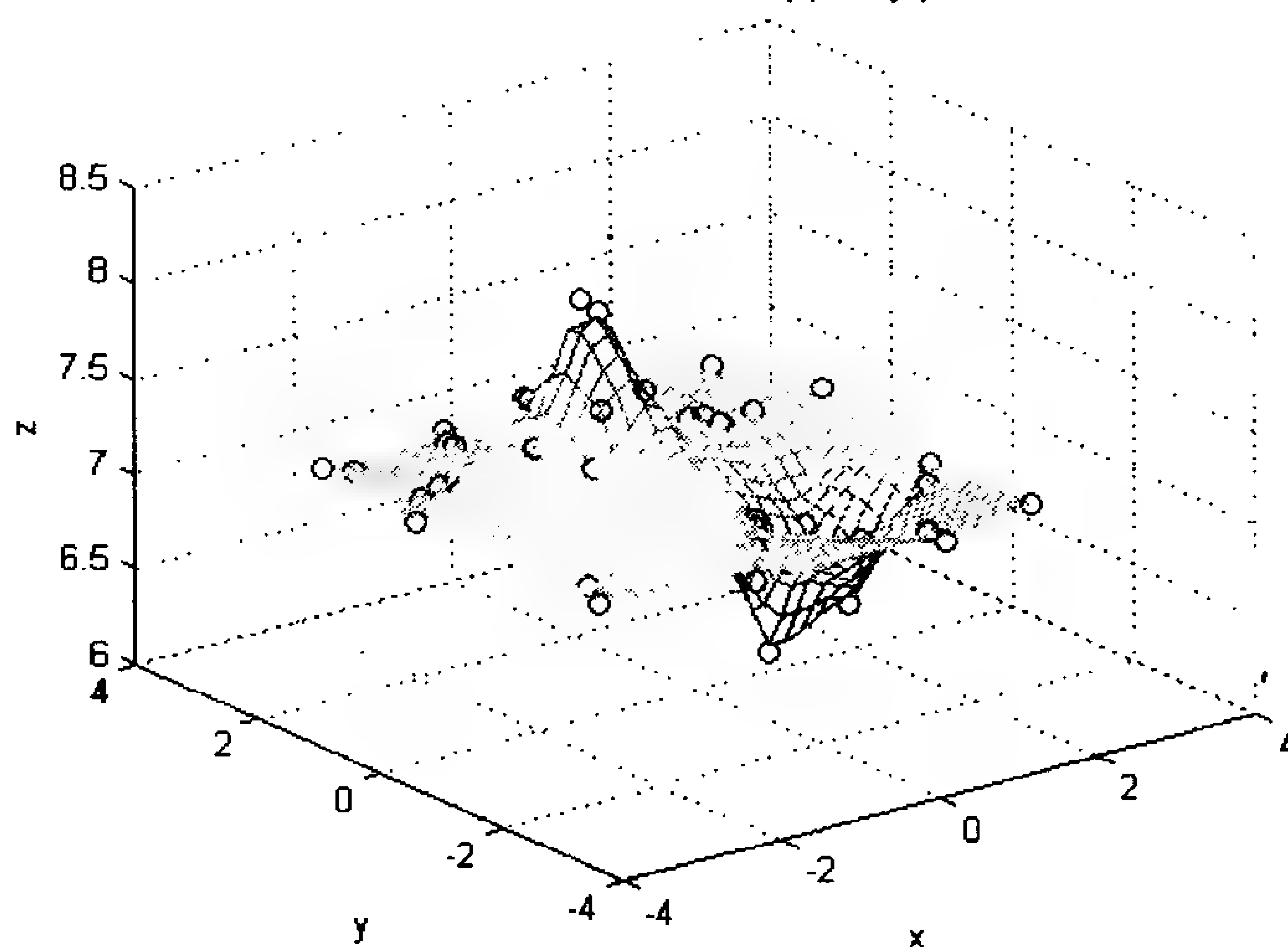
```
Y = -2.5 +(3.5 -( -2.5)) * y;% 利用 y 生成的随机变量.
Z =7 -3 * X.^3 .* exp( -X.^2 - Y.^2);% 在每个随机点(X,Y)处计算 Z 的值.
X1 = -3:0.2:3;
Y1 = -2.5:0.2:3.5;
[XI,YI] = meshgrid(X1,Y1);    % 将坐标(XI,YI)网格化.
ZI =griddata(X,Y,Z,XI,YI, 'linear') % 计算在每个插值点(XI,YI)处的插值 ZI.
mesh(XI,YI,ZI)              % 作二元拟合图形.
xlabel('x'), ylabel('y'), zlabel('z'),
title('用三角基线性内插法拟合函数 z =7 -3 x^3 exp( -x^2 - y^2) 的曲面和节点的图形')
% legend('拟合曲面','节点(xi,yi,zi)')
hold on                     % 在当前图形上添加新图形.
plot3(X,Y,Z, 'bo')          % 用蓝色小圆圈画出每个节点(X,Y,Z).
hold of                     % 结束在当前图形上添加新图形.
```

运行后屏幕显示用三角基线性内插法拟合函数 $z=7-3x^3e^{-x^2-y^2}$ 在两组不同节点处的曲面和节点的图形(见图 7 – 16)及其插值 z_i(略). 由图 7 – 16 可见,拟合曲面的峰和谷的个数不随节点值变化,但曲面的形状随节点值变化,并且拟合曲面具有突变性,曲面不光滑,这些突变点分别产生在不连续点和一阶导数不连续点. 但是拟合曲面凹凸性相对稳定,比用最近邻内插法拟合的曲面相对光滑.

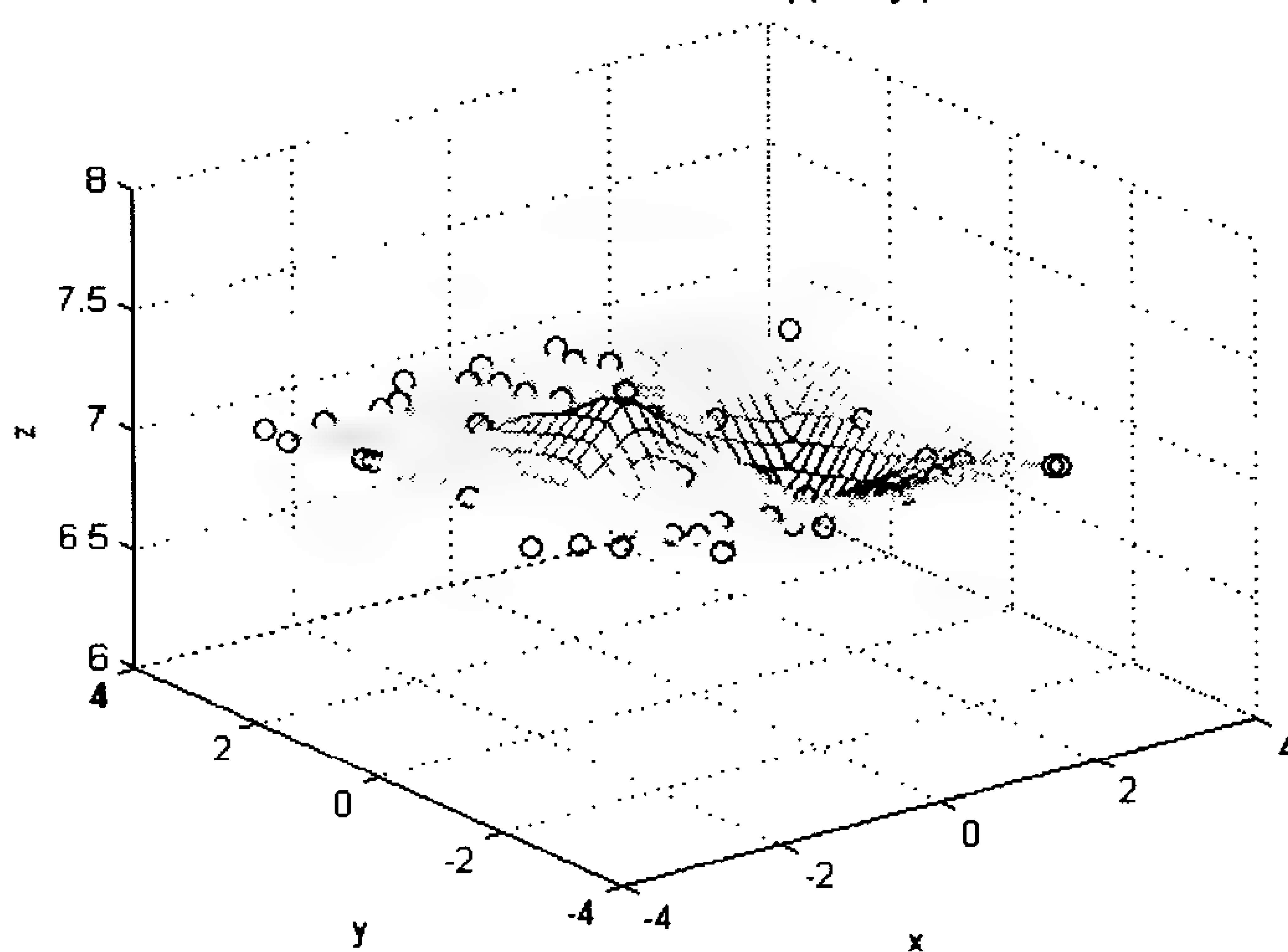
(3) 三角基三次内插法.

输入程序

```
>> x = rand(50,1);
y = rand(50,1);% 生成 50 个 1 元均匀分布随机数 x 和 y, x,y .
X = -3 +(3 -( -3)) * x;% 利用 x 生成的随机变量.
Y = -2.5 +(3.5 -( -2.5)) * y;  % 利用 y 生成的随机变量.
Z =7 -3 * X.^3 .* exp( -X.^2 - Y.^2); % 在每个随机点(X,Y)处计算 Z 的值.
X1 = -3:0.2:3;
Y1 = -2.5:0.2:3.5;
[XI,YI] = meshgrid(X1,Y1);    % 将坐标(XI,YI)网格化.
ZI =griddata(X,Y,Z,XI,YI, 'cubic') % 计算在每个插值点(XI,YI)处的插值 ZI.
mesh(XI,YI, ZI)             % 作二元拟合图形.
xlabel('x'), ylabel('y'), zlabel('z'),
title('用三角基三次内插法拟合函数 z =7 -3 x^3 exp( -x^2 - y^2) 的曲面和节点的图形')
% legend('拟合曲面','节点(xi,yi,zi)')
hold on                     % 在当前图形上添加新图形.
```



(a)

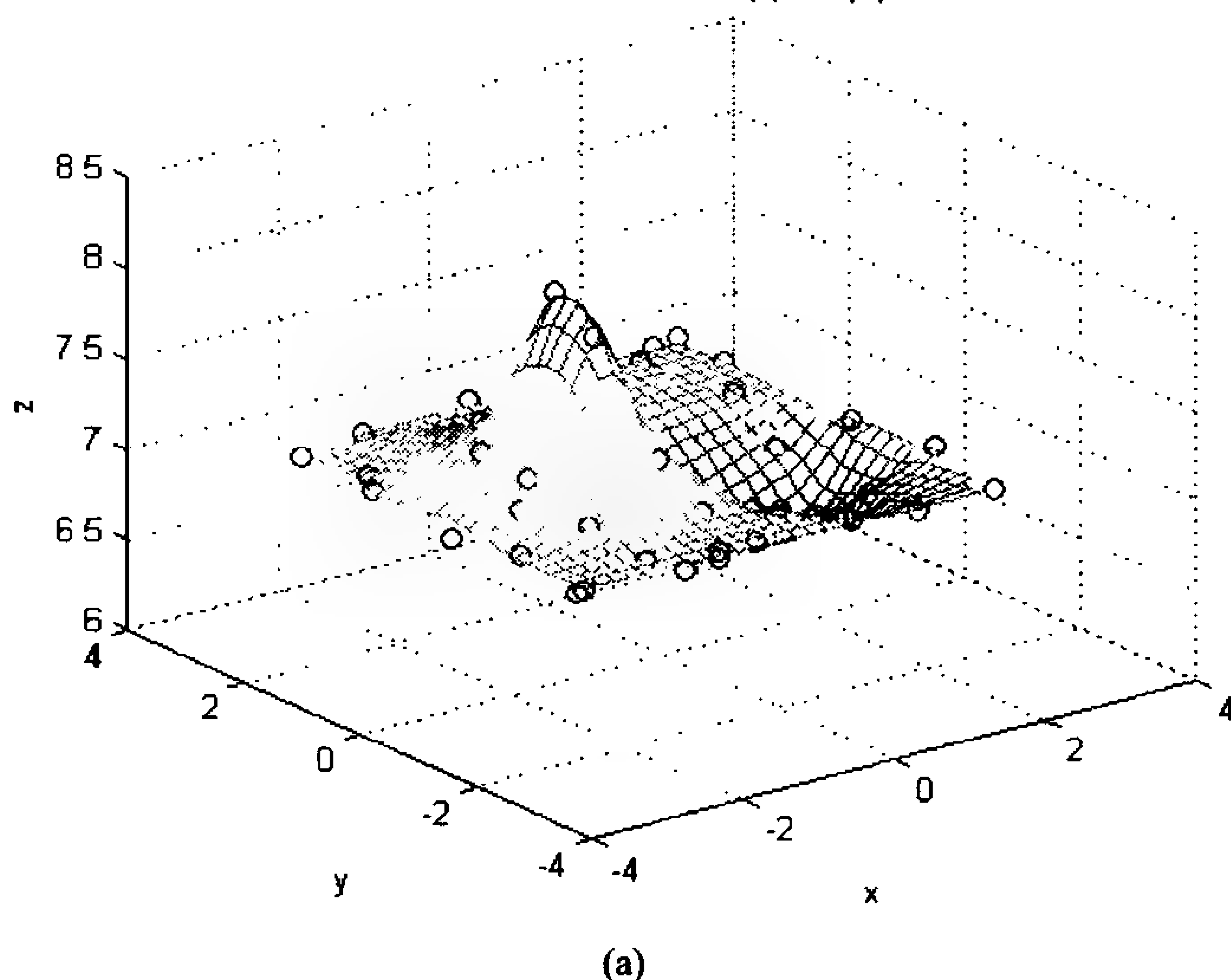


(b)

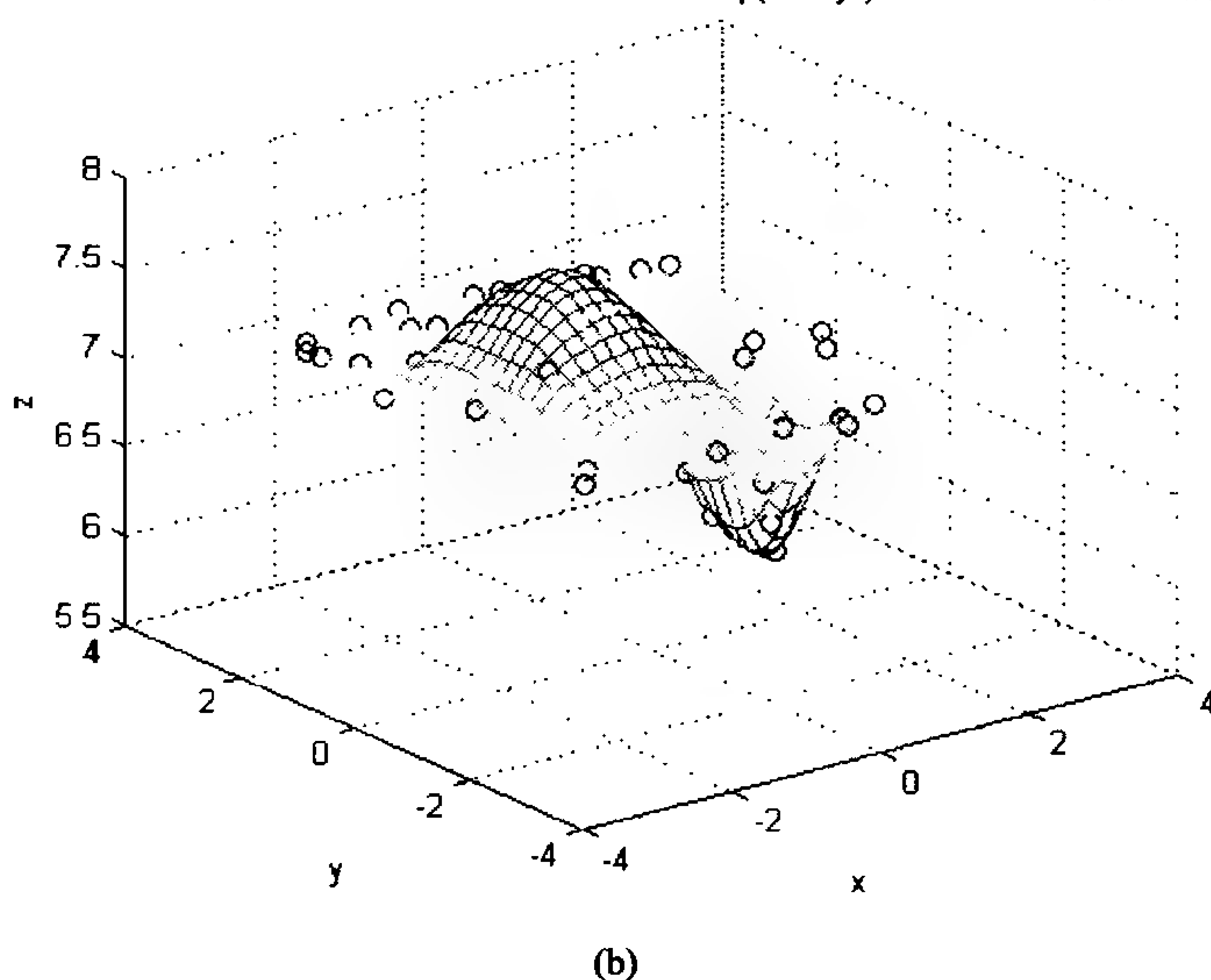
图 7－16 用三角基线性内插法拟合函数 $z=7-3x^3e^{-x^2-y^2}$ 的曲面和节点的图形

```
plot3(X,Y,Z,'bo')          % 用蓝色小圆圈画出每个节点(X,Y,Z).
hold of                    % 结束在当前图形上添加新图形.
```

运行后屏幕显示用三角基三次内插法拟合函数 $z=7-3x^3e^{-x^2-y^2}$ 在两组不同节点处的曲面和节点的图形(见图 7-17)及其插值 z_i(略). 由图 7-17 可见,拟合曲面的峰和谷的个数不随节点值变化,曲面光滑,凹凸性稳定,没有突变性,但曲面的形状随节点值变化,比用最近邻内插法和三角基线性内插法拟合的曲面相对光滑.



(a)



(b)

图 7-17 用三角基三次内插法拟合函数 $z=7-3x^3e^{-x^2-y^2}$ 的曲面和节点的图形

(4) MATLAB 4 网格化坐标方法.

输入程序

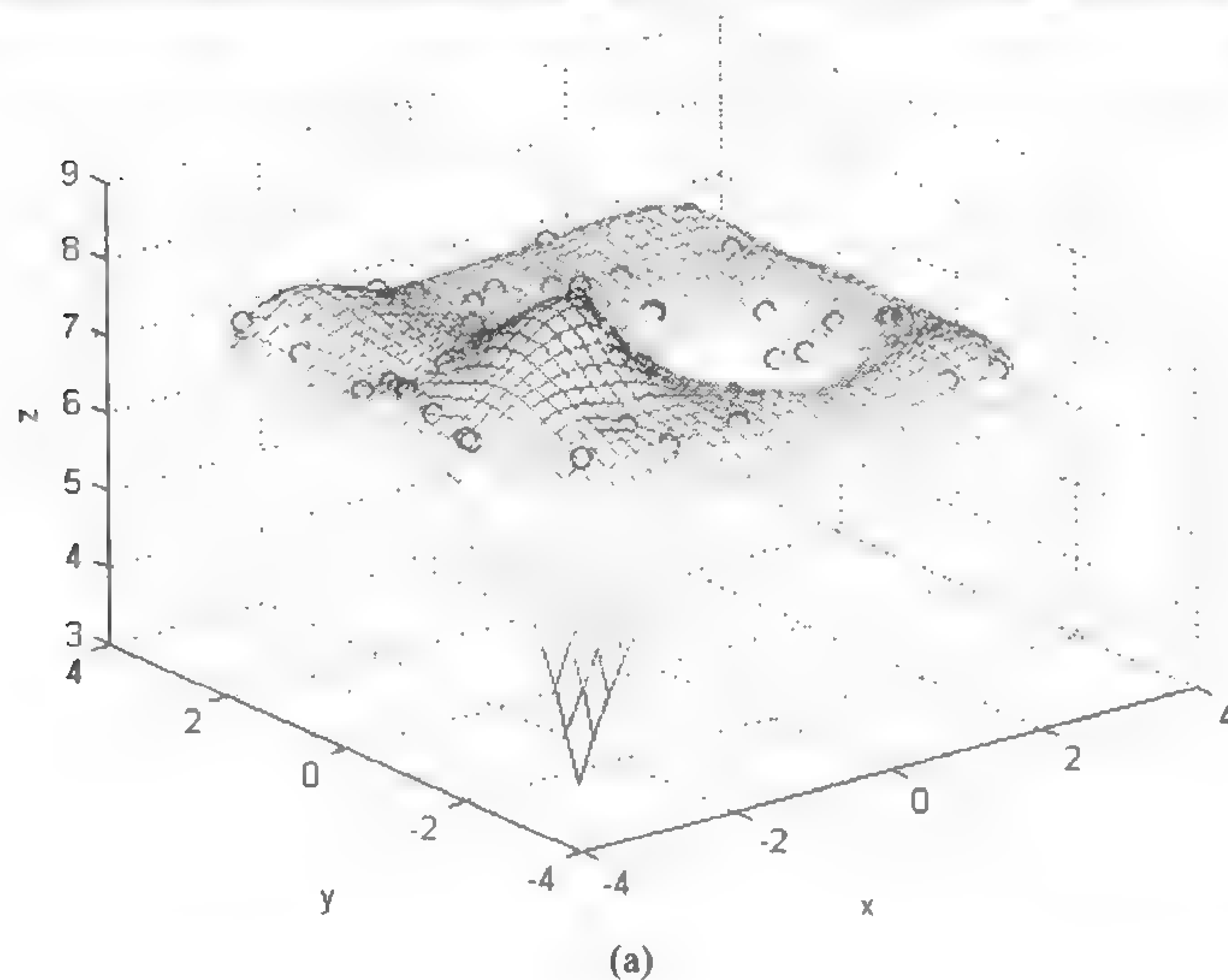
```
>> x=rand(50,1);
y=rand(50,1); %生成50个1元均匀分布随机数x和y, x,y .
X=-3+(3-(-3))*x;%利用x生成的随机变量.
Y=-2.5+(3.5-(-2.5))*y;%利用y生成的随机变量.
Z=7-3* X.^3 .* exp(-X.^2 - Y.^2);%在每个随机点(X,Y)处计算Z的值.
X1=-3:0.2:3;Y1=-2.5:0.2:3.5;
[XI,YI] = meshgrid(X1,Y1);     %将坐标(XI,YI)网格化.
ZI=griddata(X,Y,Z,XI,YI,'v4') %计算在每个插值点(XI,YI)处的插值ZI.
mesh(XI,YI, ZI)              %作二元拟合图形.
xlabel('x'), ylabel('y'), zlabel('z'),
title('用MATLAB 4网格化坐标方法拟合函数 z =7-3x^3 exp(-x^2 - y^2)
的曲面和节点的图形')
%legend('拟合曲面','节点(xi,yi,zi)')
hold on                      %在当前图形上添加新图形.
plot3(X,Y,Z,'bo')            %用蓝色小圆圈画出每个节点(X,Y,Z).
hold of                      %结束在当前图形上添加新图形.
```

运行后屏幕显示用MATLAB 4网格化坐标方法拟合函数 $z=7-3x^3e^{-x^2-y^2}$ 在两组不同节点处的曲面和节点的图形(见图7-18)及其插值 z_i(略). 由图7-18可见,拟合曲面的峰和谷的个数随节点值稍有变化,曲面光滑,没有突变性,但曲面的形状随节点值稍有变化,比用最近邻内插法和三角基线性内插法拟合的曲面光滑. 但是与被拟合曲面的误差很大,失去使用价值.

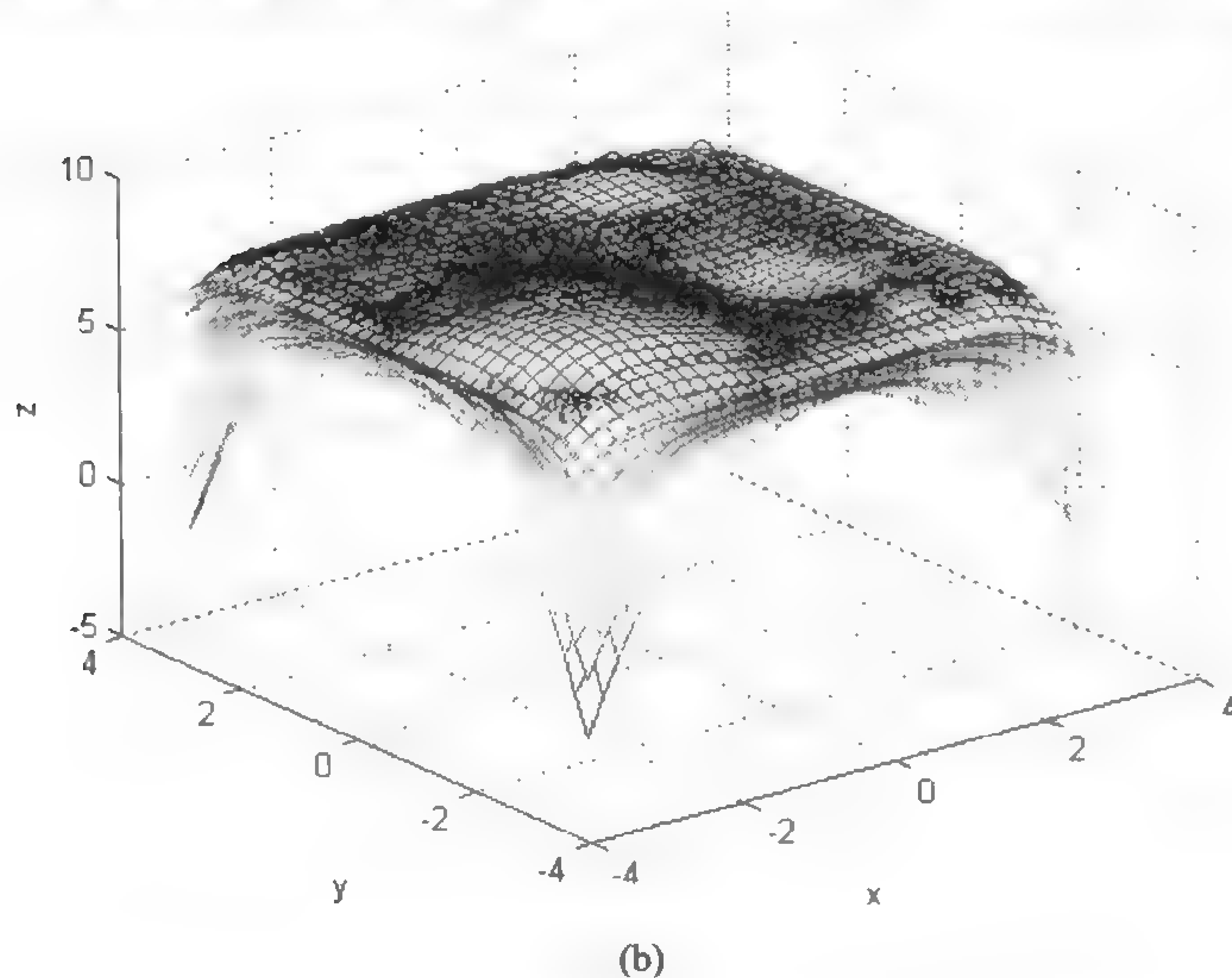
(5) 作被拟合曲面 $z=7-3x^3e^{-x^2-y^2}$ 和节点的图形.

输入程序

```
>> x=rand(50,1);
y=rand(50,1); %生成50个1元均匀分布随机数x和y, x,y .
X=-3+(3-(-3))*x;%利用x生成的随机变量.
Y=-2.5+(3.5-(-2.5))*y;  %利用y生成的随机变量.
Z=7-3* X.^3 .* exp(-X.^2 - Y.^2);%在每个随机点(X,Y)处计算Z的值.
X1=-3:0.1:3;
Y1=-2.5:0.1:3.5;
[XI,YI]=meshgrid(X1,Y1);     %将坐标(XI,YI)网格化.
ZI=7-3* XI.^3 .* exp(-XI.^2 - YI.^2);
mesh(XI,YI, ZI)              %作二元拟合图形.
xlabel('x'), ylabel('y'), zlabel('z'),
title('被拟合函数 z =7-3 x^3 exp(-x^2 - y^2) 的曲面和节点的图形')
```



(a)



(b)

图 7－18 用 MATLAB 4 网格化坐标方法拟合函数 $z=7-3x^3e^{-x^2-y^2}$的曲面和节点的图形

```
% legend('被拟合函数曲面','节点(xi,yi,zi)')
hold on                              % 在当前图形上添加新图形.
```

```
plot3(X,Y,Z,'bo')          % 用蓝色小圆圈画出每个节点(X,Y,Z).
hold of                    % 结束在当前图形上添加新图形.
```

运行后屏幕显示被拟合函数 $z=7-3x^3e^{-x^2-y^2}$ 的曲面和节点的图形(见图 7-19)及其函数值 z_i(略).

将在 (x_i, y_i) 处的二元拟合方法中最近邻内插法、三角基线性内插法、三角基三次内插法和 MATLAB 4 网格化坐标方法作出拟合曲面的图形(图 7-15 至图 7-18)与被拟合曲面图(图 7-19)进行比较,显然,定义的拟合数据的曲面的类型中,最近邻内插法和三角基线性内插法产生有突变点的曲面,但产生的拟合曲面与被拟合曲面的形态有相似之处(参见图 7-15,图 7-16 和图 7-19);MATLAB 4 网格化坐标方法和三角基三次内插法产生光滑曲面,三角基三次内插法产生的拟合曲面与被拟合曲面的形态类似,拟合误差较小(参见图 7-17 和图 7-19),而 MATLAB 4 网格化坐标方法产生的拟合曲面与被拟合曲面的形态差别很大,拟合误差最大(参见图 7-18 和图 7-19).

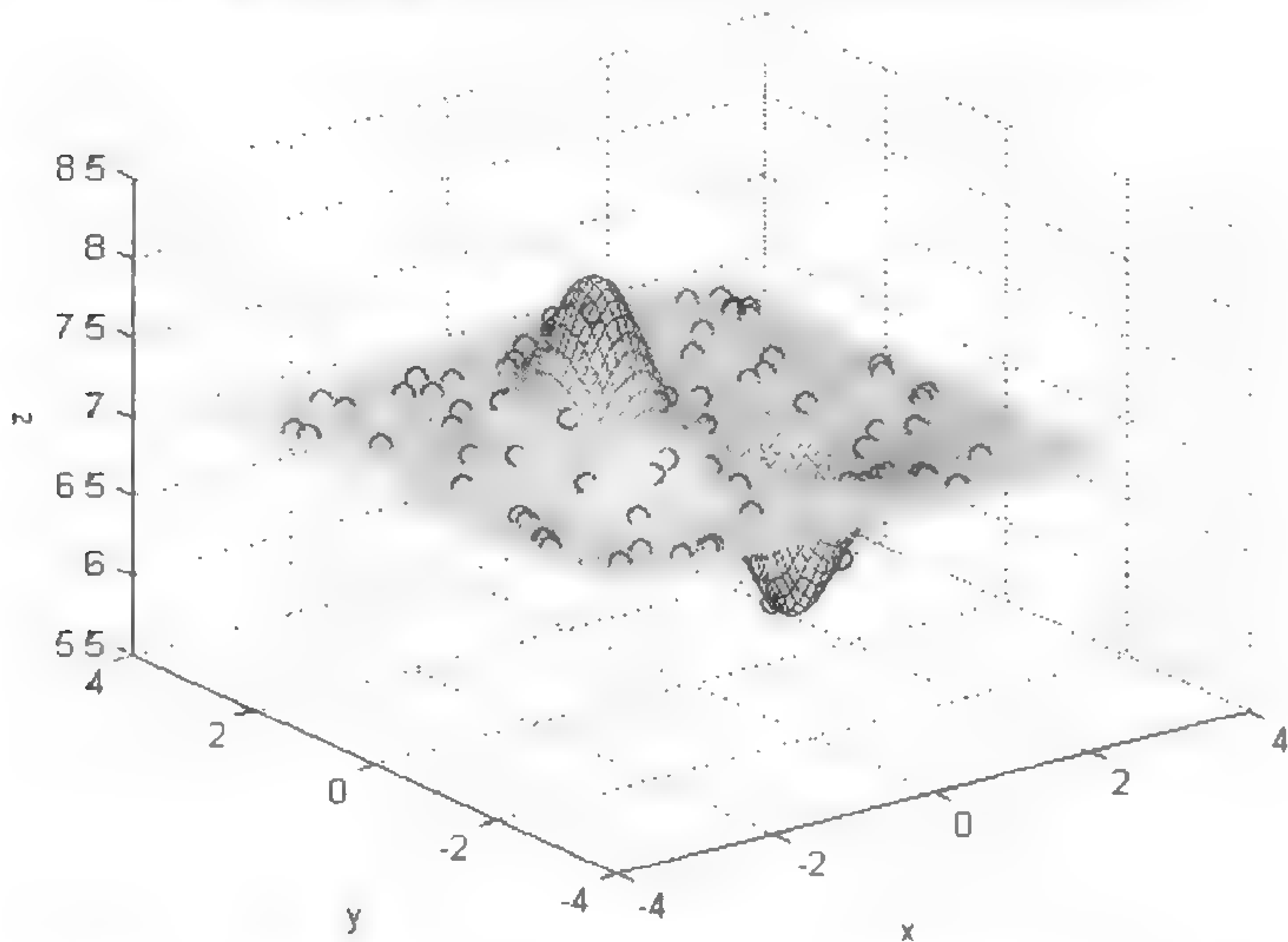
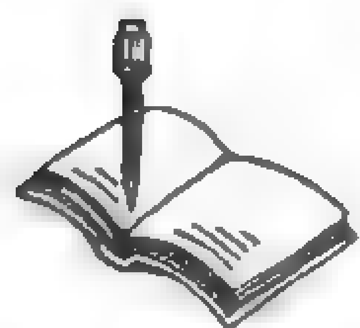


图 7-19 被拟合函数 $z=7-3x^3e^{-x^2-y^2}$ 的曲面和节点的图形



习 题 7.8

1. 设节点 (X,Y,Z) 中的 X 和 Y 分别是在区间 $[-3,3]$ 和 $[-2.5,3.5]$ 上的 50 个随机数,

Z 是函数 $z=3xye^{-x^2-y^2}$ 在(X,Y)的值，拟合点(X_I,Y_I)中的 $X_I=-3:0.2:3$，$Y_I=-2.5:0.2:3.5$. 分别用二元拟合方法中最近邻内插法、三角基线性内插法、三角基三次内插法和 MATLAB 4 网格化坐标方法计算在(X_I,Y_I)处的值，作出它们的图形，并与被拟和曲面进行比较.

2. 设节点(X,Y,Z)中的 X 和 Y 分别是在区间$[-3,3]$和$[-2.5,3.5]$上的50个随机数，Z 是函数 $z=5-2x^2e^{-x^2-y^2}$ 在(X,Y)的值，拟合点(X_I,Y_I)中的 $X_I=-3:0.2:3$，$Y_I=-2.5:0.2:3.5$. 取两组不同的节点，分别用二元拟合方法中最近邻内插法、三角基线性内插法、三角基三次内插法和 MATLAB 4 网格化坐标方法计算在(X_I,Y_I)处的值，作出它们的图形，并与被拟和曲面进行比较.

7.9 随机数据点上的 n 元拟合及其 MATLAB 程序

可以将随机数据点上的二元拟合推广到三元拟合. 所谓拟合是在空间区域 Ω 上，或者在 Ω 的分片子域 Ω_{ijk} 上，构造与被插值函数 $u=f(x,y,z)$ 的随机数据点(x_i,y_i,z_i,u_i) $(i=1,2,\cdots,n)$最为接近的较简单函数 $P(x,y,z)$，拟合函数 $P(x,y,z)$不必要过所有的随机数据点(x_i,y_i,z_i,u_i) $(i=1,2,\cdots,n)$，使得简单函数 $P(x,y,z)$最佳逼近被插值函数$f(x,y,z)$（见下面的图形），这也是随机数据点上的三元拟合的主要任务. 在 MATLAB 系统中，为我们提供了计算随机数据点上的三元和三元以上的超曲面拟合的程序 griddata3.m.此程序主要用于将随机数据点上的数据变换网格坐标后的超曲面拟合. 程序 griddata3 的功能是计算常用的三元和三元以上的拟合方法中最近邻内插法(Nearest neighbor interpolation)、三角基线性内插法(Triangle-based linear interpolation)(缺省值). 计算时输入的节点$(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$中的向量 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$，$\boldsymbol{y}=(y_1,y_2,\cdots,y_n)$，$\boldsymbol{z}=(z_1,z_2,\cdots,z_n)$的元素通常不必是单调排列，用网格化命令[X,Y,Z] = meshgrid(x,y,z)将 $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}$ 化为三元网格坐标 X,Y,Z，然后调用 griddata3 程序计算.该程序有几种调用格式如表 7-9.

表 7-9 随机数据点上的三元拟合的 MATLAB 命令

随机数据点上的三元拟合的 MATLAB 命令 griddata3	功　　能
W=griddata3(X, Y, Z, V, XI, YI, ZI)	此命令的主要功能是通常对于随机点向量$(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z},V)$中的数据拟合函数 $W=F(X,Y,Z)$的超曲面. griddata3 在由(X_i,Y_i,Z_i)产生的 W 处内插超曲面，并且该曲面总是经过数据点. 其中(X_i,Y_i,Z_i)是由 meshgrid 命令产生的均匀网格坐标.

续表

随机数据点上的 三元拟合的 MATLAB 命令 griddata3	功　　能
[…] = griddata3(…,' method ')	[…] = griddata3(…,' method ')的主要功能是用户指定三元内插法的方法.用户不输入具体的方法时,按三角基线性内插法计算.' method '可用以下方法替换: ' nearest '　表示三元最近邻内插法; ' linear '　表示三角基线性内插法(缺省值). 定义的' nearest '和' linear '方法拟合数据的超曲面都是基于一种数据的 delaunay 三角剖分.

例 7.9.1　首先利用 MATLAB 函数 rand 产生随机数据 X_1,Y_1,Z_1,然后用线性变换 $u=at+b$ (其中 $a=5,b=-5$)将随机数据 X_1,Y_1,Z_1变换为节点坐标 (X,Y,Z),再用函数 $w=7-3x^3y(z+1)e^{-x^2-y^2-z^2}$生成数据 W,用三元最近邻内插法方法计算函数 w 在插值点 $x_i=-3:0.5:10,y_i=-2:0.5:13,z_i=y_i$处拟合数据的值,并作其图形.

解　输入程序

```
>> X1 = -5 +5 * rand(10,1);
Y1 = -5 +5 * rand(10,1);
Z1 = Y1;
[X,Y,Z] = meshgrid(X1,Y1,Z1);
W = 7 -3 * X.^3 .* Y.*(Z+1).* exp( -X.^2 - Y.^2 - Z.^2);
xi = -3:0.5:10;
yi = -2:0.5:13;
zi = yi;
[XI,YI,ZI] = meshgrid(xi,yi,zi);
W1 = griddata3(X, Y, Z, W, XI, YI, ZI, 'nearest');
slice(XI,YI,ZI,W1,[ -2 4 9.5],9,[ -2 2 9]),
% shading flat
% lighting flat
xlabel('x'), ylabel('y'), zlabel('z'),
title('被拟合函数 W =7 -3X^3Y(Z+1)exp( -X^2 - Y^2 - Z^2) ');
```

```
hold on
colorbar('horiz')
view([ -30  45])
```

运行后屏幕显示三元线性拟合值(略)及其图形(如图 7-20):

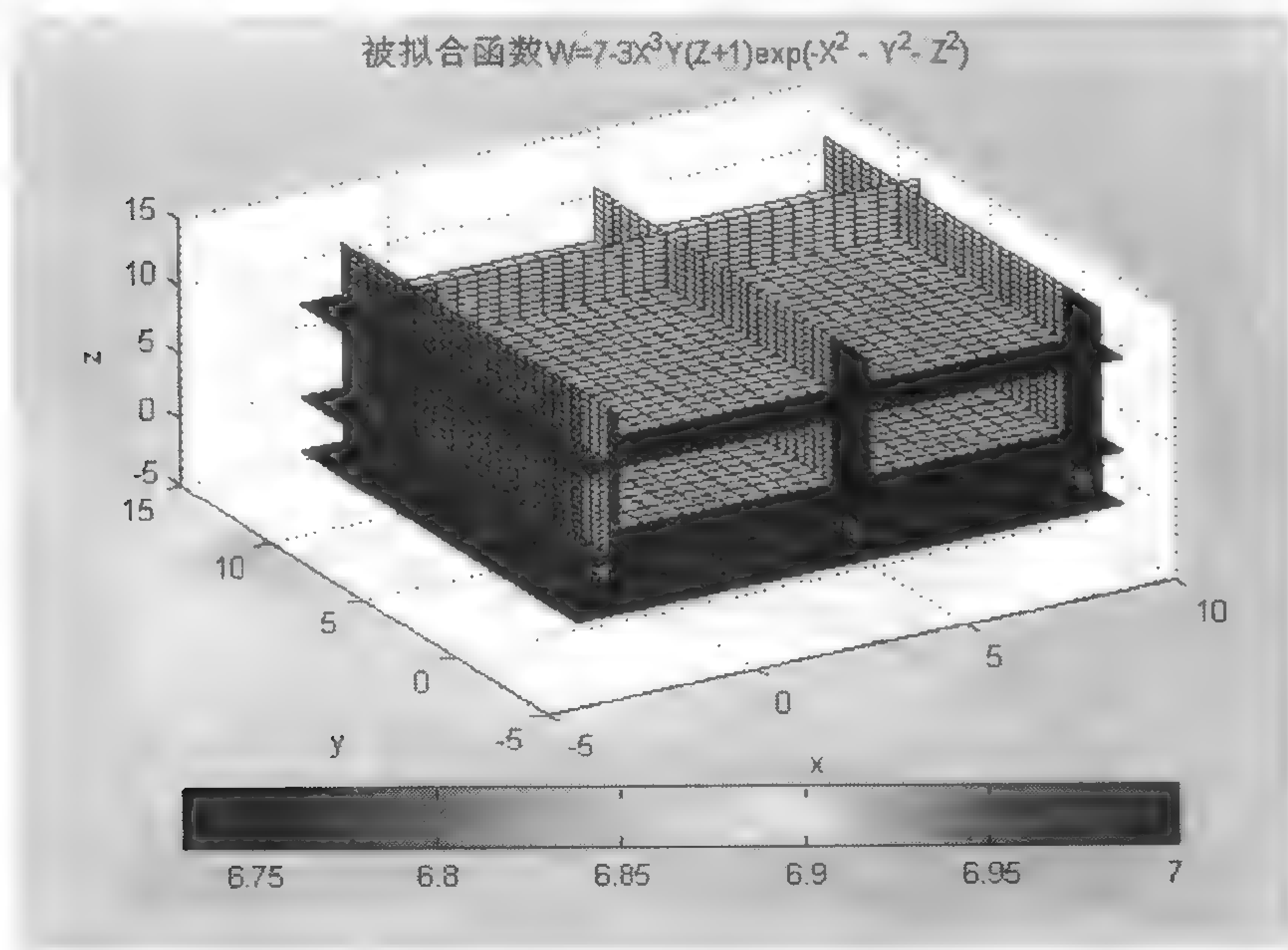


图 7-20 被拟合函数 $W=7-3x^3y(z+1)e^{-x^2-y^2-z^2}$ 的有关图形

例 7.9.2 设节点 (X,Y,Z,W) 中的 X,Y 和 Z 分别是在区间 $[-3,3]$ 和 $[-2.5,3.5]$,$Y=Z$ 上的 15 个随机数,W 是函数 $w=2+xe^{-x^2-y^2-z^2}$ 在 (X,Y,Z) 的值,拟合点 (x_i,y_i,z_i) 中的 $x_i=-3:0.2:3$,$y_i=-2.5:0.2:3.5$,$z_i=y_i$,用 'linear'方法计算拟合数据的值,并作其图形.

解 输入程序

```
>> x=rand(15,1); y=rand(15,1);
 X1 = -3 +(3 -( -3)) * x;
Y1 = -2.5 +(3.5 -( -2.5)) * y;Z1 =Y1;
[X,Y,Z] = meshgrid(X1,Y1,Z1);
W=2 +X.* exp( -X.^2 - Y.^2 - Z.^2);
xi = -3:0.2:3; yi = -2.5:0.2:3.5; zi =yi;
[X2,Y2,Z2] =meshgrid(xi,yi,zi);
W1 =griddata3(X, Y, Z, W, X2,Y2,Z2,'linear');
slice(X2,Y2,Z2,W1,[ -1 0 1.5],2,[ -2 3]),
```

```
shading flat,lighting flat,
xlabel('x'), ylabel('y'), zlabel('z'),
title('被拟合函数 W=2+X exp( -X^2 - Y^2 - Z^2)');
hold on,colorbar('horiz'),view([ -3  5])
```

运行后屏幕显示三元线性拟合值(略)及其图形(如图 7-21).

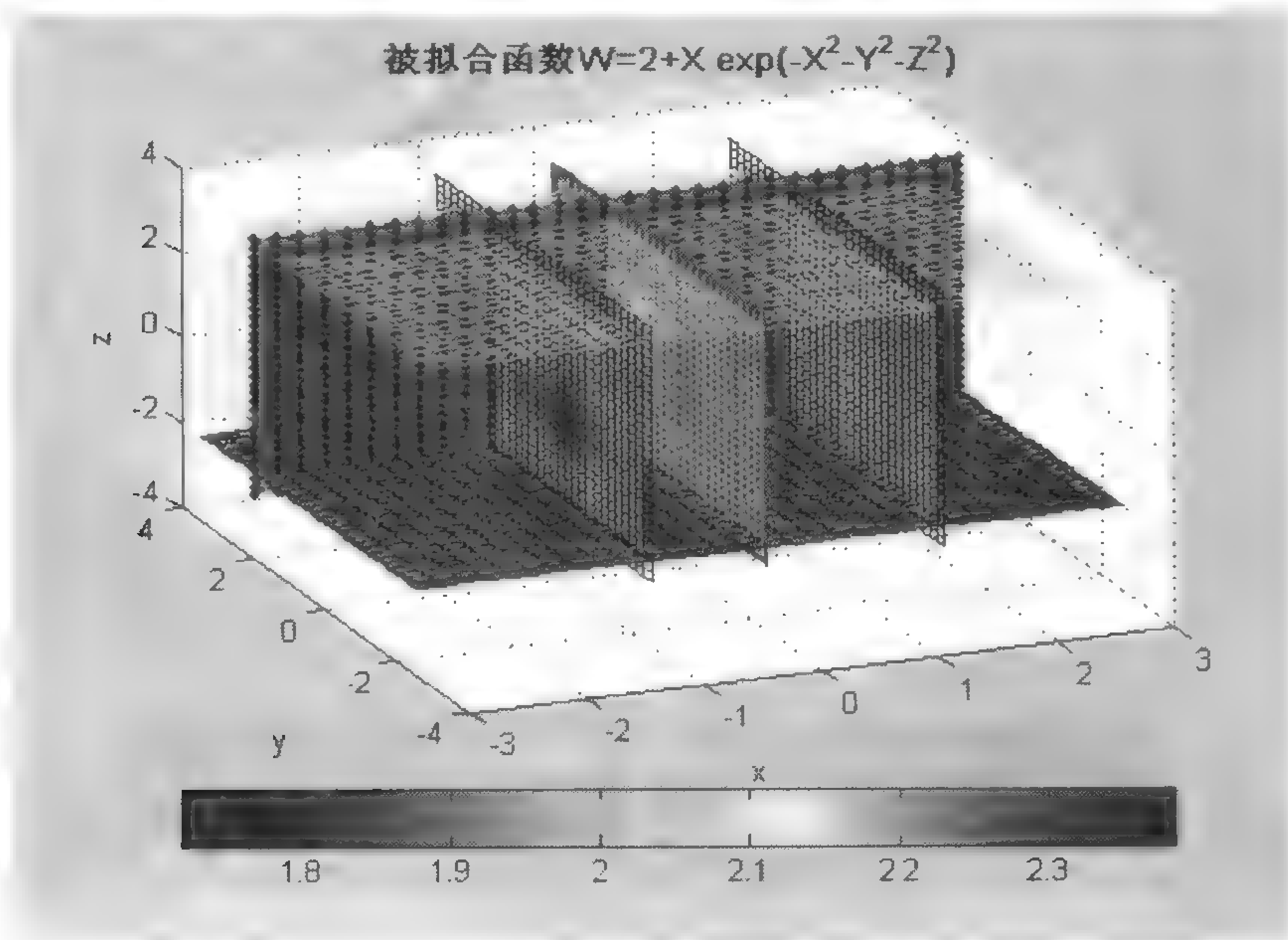
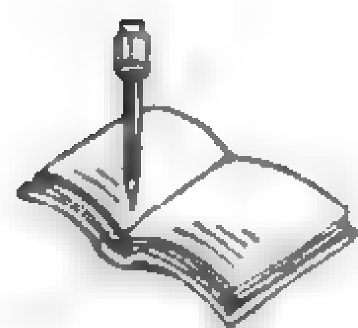


图 7-21 被拟合函数 $W=2+xe^{-x^2-y^2-z^2}$ 的有关图形



习 题 7.9

1. 设节点(X,Y,Z,W)中的X,Y和Z分别是在区间$[-3,3]$和$[-2.5,3.5]$,$Y=Z$上的15个随机数,W是函数$w=3-2xe^{-x^2-y^2-z^2}$在(X,Y,Z)的值,用' linear '方法计算函数w在插值点$x_i=-3:0.2:3$,$y_i=-2.5:0.2:3.5$,$z_i=y_i$处的拟合数据的值,并作其图形.

2. 首先利用 MATLAB 函数 rand 产生随机数据X_1,Y_1,Z_1,然后用线性变换$u=at+b$(其中$a=6,b=-7$)将随机数据X_1,Y_1,Z_1变换为节点坐标(X,Y,Z),再用函数$w=6-5x^3\cdot(y+1)e^{-x^2-y^2-z^2}$生成数据$W$,用' nearest '方法计算函数$w$在插值点$x_i=-3:0.5:10$,$y_i=-2:0.5:13$,$z_i=y_i$处的拟合数据的值,并作其图形.

第八章 数值微分

微分是我们在高等数学中学过的最基本的运算. 一方面它是一些非常重要的数学工具(微分方程、概率论等)的基础,另一方面它在实际问题中也有着许多直接的应用. 函数的导数计算可分为数值求导和符号求导两类方法. 数值求导是导函数在某点处的值的一类近似计算的方法,例如遇到由离散数据或者图形表示的函数就只有求助于数值方法. 符号求导是指对给定的函数求它在某区域上的导函数或在某点处的导数. 二者在概念上有着密切的联系,但在求解思路上却有明显差别. 本章重点介绍数值微分及其用 MATLAB 计算的方法,为了估计数值微分计算的误差,在第一节简单介绍用 MATLAB 计算符号微分的方法.

8.1 用 MATLAB 求各阶(偏)导数和(全)微分

在 MATLAB 的函数库中有符号求导的程序 `diff.m`,可以调用此程序求符号(偏)导数和(全)微分,不但使用方便,而且计算准确、迅速,尤其是求结构复杂的高阶(偏)导数和(全)微分更显示出其优越性. 在本节主要介绍用 `diff.m` 求一(多)元函数的各阶(偏)导函数和(全)微分及其在某点处的各阶(偏)导数和(全)微分.

8.1.1 用 MATLAB 符号计算一元函数的导数和微分

在大学数学中,一元函数 $y=f(x)$ 的各阶导函数记为 y',y'',y''',…,在这里分别记作 $yx,yxx,yxxx$,…. 用 `diff` 求符号导数、微分和在一点处的导数和微分的调用格式和功能如表 8-1 和例题所示.

表 8-1 求一元函数 $y=f(x)$ 的各阶导函数的 MATLAB 方法

符号求导的命令	功 能
`yx = diff( f(x),x)`	求函数 $y=f(x)$ 对 x 的一阶导函数 $y'=f'(x)$
`dy = diff( f(x),x) * dx`	求函数 $y=f(x)$ 的一阶微分 $dy=f'(x)dx$
`yxx = diff( f(x),x,2)` 或 `yxx = diff(yx,x)`	求函数 $y=f(x)$ 对 x 的二阶导函数 $y''=f''(x)$
`yxxx = diff( f(x),x,3)` 或 `yxxx = diff(yxx,x)`	求函数 $y=f(x)$ 对 x 的三阶导函数 $y'''=f'''(x)$

续表

符号求导的命令	功 能
yxn = diff(f(x),x,n)	求函数 $y=f(x)$ 对 x 的 n 阶导函数 $y^{(n)}=f^{(n)}(x)$
yn = simple(yxn)	将 n 阶导函数 $y^{(n)}=f^{(n)}(x)$ 化简,并记作 yn.
pretty(diff(f(x),x)	输出一个符合日常书写习惯的一阶导函数的表达式

例 8.1.1 设一元函数 $y=\arctan(3x^2+2\sin(5x^3-2))$.

(1) 求 y 的一阶导数和四阶导数,并化简;(2) 求 y 的一阶微分.

解 (1) 计算 y 对 x 的一阶导数、一阶微分和四阶导数,输入程序

```
>> syms x dx
y = atan(3*x^2+2*sin(5*x^3-2));
yx = diff(y,x);dy = yx*dx,pretty(dy)
yx4 = diff(y,x,4);y1 = simple(yx), pretty(y1),y4 = simple(yx4),
```

运行后屏幕显示 y 对 x 的一阶导数 $y1$、一阶微分 dy 和四阶导数 $y4$ 如下

```
dy
(6*x+30*cos(5*x^3-2)*x^2)/(1+(3*x^2+2*sin(5*x^3-2))^2)*dx
y1 =
(6*x+30*cos(5*x^3-2)*x^2)/(1+(3*x^2+2*sin(5*x^3-2))^2)
y4 =
(101250*sin(5*x^3-2)*x^8-81000*cos(5*x^3-2)*x^5-9000*sin(5*x^3-2)*x^2)/(1+(3*x^2+2*sin(5*x^3-2))^2)-8*(-6750*cos(5*x^3-2)*x^6-2700*sin(5*x^3-2)*x^3+60*cos(5*x^3-2))/(1+(3*x^2+2*sin(5*x^3-2))^2)^2*(3*x^2+2*sin(5*x^3-2))*(6*x+30*cos(5*x^3-2)*x^2)+48*(6-450*sin(5*x^3-2)*x^4+60*cos(5*x^3-2)*x)/(1+(3*x^2+2*sin(5*x^3-2))^2)^3*(3*x^2+2*sin(5*x^3-2))^2*(6*x+30*cos(5*x^3-2)*x^2)^2-12*(6-450*sin(5*x^3-2)*x^4+60*cos(5*x^3-2)*x)/(1+(3*x^2+2*sin(5*x^3-2))^2)^2*(6*x+30*cos(5*x^3-2)*x^2)^2-6*(6-450*sin(5*x^3-2)*x^4+60*cos(5*x^3-2)*x)^2/(1+(3*x^2+2*sin(5*x^3-2))^2)^2*(3*x^2+2*sin(5*x^3-2))-48*(6*x+30*cos(5*x^3-2)*x^2)^4/(1+(3*x^2+2*sin(5*x^3-2))^2)^4*(3*x^2+2*sin(5*x^3-2))^3+24*(6*x+30*cos(5*x^3-2)*x^2)^4/(1+(3*x^2+2*sin(5*x^3-2))^2)^3*(3*x^2+2*sin(5*x^3-2))
```

(2) 另外还可以分别用下面几种形式的 MATLAB 程序分别计算 y 对 x 的一阶导数、一阶微分和四阶导数

```
>> syms x dx
```

```
y = atan(3 * x^2 + 2 * sin(5 * x^3 - 2));dy = diff(y,x) * dx
```

或 `dy = diff(atan(3 * x^2 + 2 * sin(5 * x^3 - 2)),x) * dx`

或
```
yx2 = diff((6 * x + 30 * cos(5 * x^3 - 2) * x^2)/(1 + (3 * x^2 + 2 * sin(5 * x^3 - 2))^2),x)
yx4 = diff((6 * x + 30 * cos(5 * x^3 - 2) * x^2)/(1 + (3 * x^2 + 2 * sin(5 * x^3 - 2))^2),x,2)
```

所得到的结果与(1)相同.

例 8.1.2 设一元函数 $y = (3x^2 + 2\cos(5x^3 - 2))\ln(13x^5 + 6) + \dfrac{2x - 1}{5x^2 + 6}$.

(1) 求 y 的五阶导数 $y^{(5)}(x)$ 和微分 $\mathrm{d}y$;

(2) 求 y 在 $x = 0$ 处的五阶导数 $y^{(5)}(0)$ 和微分 $\mathrm{d}y\Big|_{x=0}$.

解 (1) 输入计算 y 对 x 的一阶微分和五阶导数的程序

```
>> syms x dx
y = (3 * x^2 + 2 * cos(5 * x^3 - 2)) * log(13 * x^5 + 6) + (2 * x - 1)/(5 * x^2 + 6);
dy = diff (y,x) * dx,yx5 = diff(y,x,5);y5 = simple(yx5)
```

运行后屏幕显示化简后的 y 对 x 的五阶导数 $y^{(5)}(x)$ 和 y 的微分 $\mathrm{d}y$ (略).

(2) 计算 y 在 $x = 0$ 处的五阶导数 $y^{(5)}(0)$ 和微分 $\mathrm{d}y\Big|_{x=0}$,输入程序

```
>> syms dx,x = 0;
dy = ((6 * x - 30 * sin(5 * x^3 - 2) * x^2) * log(13 * x^5 + 6) + 65 * (3 * x
^2 + 2 * cos(5 * x^3 - 2)) * x^4/(13 * x^5 + 6) + 2/(5 * x^2 + 6) - 10 * (2 * x -
1)/(5 * x^2 + 6)^2 * x) * dx
y5 = 6000/(5 * x^2 + 6)^3 + 7800 * (6 * x - 30 * sin(5 * x^3 - 2) * x^2) *
x/(13 * x^5 + 6)  90000 * (2 * x - 1)/(5 * x^2 + 6)^4 * x - 4284150000 * (3 * x^2
+ 2 * cos(5 * x^3 - 2)) * x^15/(13 * x^5 + 6)^4 + 325 * (101250 * cos(5 * x^3 -
2) * x^8 + 81000 * sin(5 * x^3 - 2) * x^5 - 9000 * cos(5 * x^3 - 2) * x^2) * x^4/
(13 * x^5 + 6) + ( - 1518750 * x^10 * sin(5 * x^3 - 2) + 2025000 * cos(5 * x^3 -
2) * x^7 + 540000 * sin(5 * x^3 - 2) * x^4 - 18000 * cos(5 * x^3 - 2) * x) *
log(13 * x^5 + 6) - 42250 * (6750 * sin(5 * x^3 - 2) * x^6 - 2700 * cos(5 * x^3 -
2) * x^3 - 60 * sin(5 * x^3 - 2)) * x^8/(13 * x^5 + 6)^2 + 5492500 * (6 - 450 *
cos(5 * x^3 - 2) * x^4 - 60 * sin(5 * x^3 - 2) * x) * x^12/(13 * x^5 + 6)^3 -
12000000 * (2 * x - 1)/(5 * x^2 + 6)^6 * x^5 + 65910000 * (6 * x - 30 * sin(5 *
x^3 - 2) * x^2) * x^11/(13 * x^5 + 6)^3 + 197730000 * (3 * x^2 + 2 * cos(5 * x^3
- 2)) * x^10/(13 * x^5 + 6)^3 - 535518750 * (6 * x - 30 * sin(5 * x^3 - 2) * x^2)
* x^16/(13 * x^5 + 6)^4 + 27846975000 * (3 * x^2 + 2 * cos(5 * x^3 - 2)) * x^20/
(13 * x^5 + 6)^5 + 2400000 * (2 * x - 1)/(5 * x^2 + 6)^5 * x^3 + 1560 * (3 * x^2 +
2 * cos(5 * x^3 - 2))/(13 * x^5 + 6) - 2535000 * (3 * x^2 + 2 * cos(5 * x^3 - 2))
* x^5/(13 * x^5 + 6)^2 - 507000 * (6 - 450 * cos(5 * x^3 - 2) * x^4 - 60 * sin(5
```

```
*x^3 2)*x)*x^7/(13*x^5+6)^2+7800*(6-450*cos(5*x^3-2)*x^4-
60*sin(5*x^3-2)*x)*x^2/(13*x^5+6)-2028000*(6*x-30*sin(5*x^
3-2)*x^2)*x^6/(13*x^5+6)^2+2600*(6750*sin(5*x^3-2)*x^6-
2700*cos(5*x^3-2)*x^3-60*sin(5*x^3-2))*x^3/(13*x^5+6)-
360000/(5*x^2+6)^4*x^2+2400000/(5*x^2+6)^5*x^4
```

运行后屏幕显示 y 在 $x=0$ 处的五阶导数 $y^{(5)}(0)$ 和微分 $dy|_{x=0}$ 如下

```
dy =
    1/3*dx
y5 =
    -188.6186
```

8.1.2 用 MATLAB 符号计算多元函数的偏导数和全微分

我们不但可以调用 diff 程序作一元函数的符号求导,而且还可以作多元函数的符号求偏导数.在这里我们将二元函数 $z=f(x,y)$ 的各阶偏导数分别记作 $zx,zy,zxx,zyy,zxy,zyx,zxxx,zxxy,zxyy,zyyy,\cdots$.如果二元函数 $z=f(x,y)$ 的两个混合偏导数 zyx 和 zxy 在某区域 D 上连续,则 $zyx=zxy$.同理,如果三元函数 $u=f(x,y,z)$ 的混合偏导数 $zxxy,zyxx$ 和 $zxyx$ 在某区域 D 上连续,则 $zxxy=zyxx=zxyx$.用 diff 可以作多元函数的符号求偏导函数、全微分和在一点处的偏导数和全微分,下面以二元函数和三元函数为例说明 diff 的调用格式和功能(如表 8-2 和例题所示).读者可以根据用 diff 求二元函数和三元函数的偏导数和全微分的方法推广到四元和四元以上的函数.

表 8-2 用 diff 做符号求偏导数和全微分的调用格式和功能

符号求导的命令	功 能
zx = diff(f(x,y),x)	求 $z=f(x,y)$ 对 x 的一阶偏导函数 $z'_x=f'_x(x,y)$
zy = diff(f(x,y),y)	求 $z=f(x,y)$ 对 y 的一阶偏导函数 $z'_y=f'_y(x,y)$
dz = zx * dx + zy * dy	求 $z=f(x,y)$ 的全微分 $dz=f'_x(x,y)dx+f'_y(x,y)dy$
zxx = diff(zx,x)	求 $z=f(x,y)$ 对 x 的二阶偏导函数 $z''_{xx}=f''_{xx}(x,y)$
zxy = diff(zx,y)	求 $z=f(x,y)$ 的二阶混合偏导函数 $z''_{xy}=f''_{xy}(x,y)$
zxn = diff(f(x,y),x,n)	求函数 $z=f(x,y)$ 对 x 的 n 阶偏导函数 $\frac{\partial^n z}{\partial x^n}$
zyn = diff(f(x,y),y,n)	求函数 $z=f(x,y)$ 对 y 的 n 阶偏导函数 $\frac{\partial^n z}{\partial y^n}$
ux = diff(f(x,y,z),x)	求 $u=f(x,y,z)$ 对 x 的一阶偏导函数 $u'_x=f'_x(x,y,z)$
uy = diff(f(x,y,z),y)	求 $u=f(x,y,z)$ 对 y 的一阶偏导函数 $u'_y=f'_y(x,y,z)$
uz = diff(f(x,y,z),z)	求 $u=f(x,y,z)$ 对 z 的一阶偏导函数 $u'_z=f'_z(x,y,z)$
du = ux * dx + uy * dy + uz * dz	求 $u=f(x,y,z)$ 的全微分 $du=f'_x(x,y,z)\,dx+f'_y(x,y,z)\,dy+f'_z(x,y,z)\,dz$
uyx = diff(uy,x)	求 $u=f(x,y,z)$ 的二阶混合偏导函数 $u''_{yx}=f''_{yx}(x,y,z)$

续表

符号求导的命令	功 能
uyxy =diff(uyx,y)	求 $u=f(x,y,z)$ 的三阶混合偏导函数 $u'''_{yxy}=f'''_{yxy}(x,y,z)$
Zx = -diff(F,x)/ diff(F,z) Zy = -diff(F,y)/ diff(F,z)	隐函数 $F(x,y,z)=0$ 求偏导函数 $\frac{\partial z}{\partial x},\frac{\partial z}{\partial y}$
pretty (diff(f(x,y,z),x)	输出一个符合日常书写习惯的表达式

例 8.1.3 设二元函数 $z=\arctan(3x^2+2\sin(5x^3-2y^5))+\dfrac{5x-6y-98}{7x^{13}+31y^{21}+9}$.

(1) 求 z 的一阶偏导函数和全微分;

(2) 求高阶偏导数 $\dfrac{\partial^2 z}{\partial x\partial y},\dfrac{\partial^3 z}{\partial x\partial y^2},\dfrac{\partial^2 z}{\partial y^2}$;

(3) 求 z 在 $x=0,y=1$ 处的一阶偏导函数、全微分、高阶偏导数 $\left.\dfrac{\partial^2 z}{\partial x\partial y}\right|_{\substack{x=0\\y=1}}$, $\left.\dfrac{\partial^3 z}{\partial x\partial y^2}\right|_{\substack{x=0\\y=1}}$, $\left.\dfrac{\partial^2 z}{\partial y^2}\right|_{\substack{x=0\\y=1}}$.

解 (1)输入计算 z 的一阶偏导函数和全微分,高阶偏导数 $\dfrac{\partial^2 z}{\partial x\partial y},\dfrac{\partial^3 z}{\partial x\partial y^2},\dfrac{\partial^2 z}{\partial y^2}$ 的程序

```
>> syms x y dx dy
z =atan(3*x^2 +2*sin(5*x^3 -2*y^5)) +( 5*x -6*y -98)/( 7*x^13 +31*y^21 +9);
zx =diff (z,x),pretty(diff(z,x)),zy =diff (z,y),dz =zx*dx +zy*dy,
pretty(diff(dz)),zxy =diff(zx,y),zxyy =diff(zxy,y),zyy =diff(zy,y),
```

运行后屏幕显示 z 的一阶偏导函数 z_x 和全微分 dz,高阶偏导数 z_{xy},z_{xyy},z_{yy}(略).

(2) 输入计算 z 在 $x=0,y=1$ 处的一阶偏导数,全微分,高阶偏导数 $\left.\dfrac{\partial^2 z}{\partial x\partial y}\right|_{\substack{x=0\\y=1}}$, $\left.\dfrac{\partial^3 z}{\partial x\partial y^2}\right|_{\substack{x=0\\y=1}}$, $\left.\dfrac{\partial^2 z}{\partial y^2}\right|_{\substack{x=0\\y=1}}$ 的程序

```
>> syms dx dy,x =0;y =1;
zx =(6*x +30*cos(5*x^3 -2*y^5)*x^2)/(1 +(3*x^2 +2*sin(5
*x^3 -2*y^5))^2) +5/(7*x^13 +31*y^21 +9) -91*(5*x -6*y -98)/(7*
x^13 +31*y^21 +9)^2*x^12
zy = -20*cos(5*x^3 -2*y^5)*y^4/(1 +(3*x^2 +2*sin(5*x^3 -2
*y^5))^2) -6/(7*x^13 +31*y^21 +9) -651*(5*x -6*y -98)/(7*x^13 +
31*y^21 +9)^2*y^20
dz =((6*x +30*cos(5*x^3 -2*y^5)*x^2)/(1 +(3*x^2 +2*sin(5
```

```
*x^3-2*y^5))^2)+5/(7*x^13+31*y^21+9)-91*(5*x-6*y-98)/(7*
x^13+31*y^21+9)^2*x^12)*dx+(-20*cos(5*x^3-2*y^5)*y^4/(1+
(3*x^2+2*sin(5*x^3-2*y^5))^2)-6/(7*x^13+31*y^21+9)-651*(5
*x-6*y-98)/(7*x^13+31*y^21+9)^2*y^20)*dy
    zxy=300*sin(5*x^3-2*y^5)*y^4*x^2/(1+(3*x^2+2*sin(5*
x^3-2*y^5))^2)+40*(6*x+30*cos(5*x^3-2*y^5)*x^2)/(1+(3*x^2
+2*sin(5*x^3-2*y^5))^2)^2*(3*x^2+2*sin(5*x^3-2*y^5))*cos(5
*x^3-2*y^5)*y^4-3255/(7*x^13+31*y^21+9)^2*y^20+546/(7*x^13
+31*y^21+9)^2*x^12+118482*(5*x-6*y-98)/(7*x^13+31*y^21+9)
^3*x^12*y^20
    zxyy=-3000*cos(5*x^3-2*y^5)*y^8*x^2/(1+(3*x^2+2*
sin(5*x^3-2*y^5))^2)+1200*sin(5*x^3-2*y^5)*y^3*x^2/(1+(3*x
^2+2*sin(5*x^3-2*y^5))^2)+24000*sin(5*x^3-2*y^5)*y^8*x^2/
(1+(3*x^2+2*sin(5*x^3-2*y^5))^2)^2*(3*x^2+2*sin(5*x^3-2*y
^5))*cos(5*x^3-2*y^5)+3200*(6*x+30*cos(5*x^3-2*y^5)*x^2)/
(1+(3*x^2+2*sin(5*x^3-2*y^5))^2)^3*(3*x^2+2*sin(5*x^3-2*y
^5))^2*cos(5*x^3-2*y^5)^2*y^8-800*(6*x+30*cos(5*x^3-2*y^5)
*x^2)/(1+(3*x^2+2*sin(5*x^3-2*y^5))^2)^2*cos(5*x^3-2*y^5)^
2*y^8+400*(6*x+30*cos(5*x^3-2*y^5)*x^2)/(1+(3*x^2+2*sin(5
*x^3-2*y^5))^2)^2*(3*x^2+2*sin(5*x^3-2*y^5))*sin(5*x^3-2*
y^5)*y^8+160*(6*x+30*cos(5*x^3-2*y^5)*x^2)/(1+(3*x^2+2*
sin(5*x^3-2*y^5))^2)^2*(3*x^2+2*sin(5*x^3-2*y^5))*cos(5*x^
3-2*y^5)*y^3+4238010/(7*x^13+31*y^21+9)^3*y^40-65100/(7*x^
13+31*y^21+9)^2*y^18-1421784/(7*x^13+31*y^21+9)^3*x^12*y^20
-231395346*(5*x-6*y-98)/(7*x^13+31*y^21+9)^4*x^12*y^40+
2369640*(5*x-6*y-98)/(7*x^13+31*y^21+9)^3*x^12*y^19
    zyy=-200*sin(5*x^3-2*y^5)*y^8/(1+(3*x^2+2*sin(5*x^3
-2*y^5))^2)-80*cos(5*x^3-2*y^5)*y^3/(1+(3*x^2+2*sin(5*x^3
-2*y^5))^2)-800*cos(5*x^3-2*y^5)^2*y^8/(1+(3*x^2+2*sin(5*
x^3-2*y^5))^2)^2*(3*x^2+2*sin(5*x^3-2*y^5))+7812/(7*x^13+
31*y^21+9)^2*y^20+847602*(5*x-6*y-98)/(7*x^13+31*y^21+9)^
3*y^40-13020*(5*x-6*y-98)/(7*x^13+31*y^21+9)^2*y^19
```

运行后屏幕显示计算结果如下

```
    zx =        zy =        zxy =       zxyy =      zyy =
     0.1250      44.0973     -2.0344     25.5314     -462.6399
    dz =
      1/8*dx+1551535530263059/35184372088832*dy
```

故 $dz \approx 0.1250dx + 44.0973dy$.

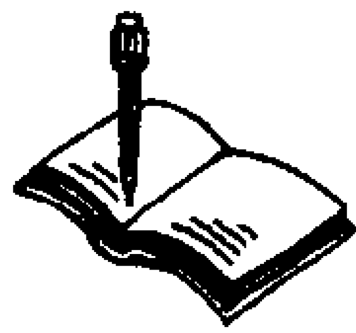
例 8.1.4 由隐函数 $2x+y+z=e^{-x-3y-2z}$ 确定 $z=z(x,y)$，求 $\frac{\partial z}{\partial y}$ 和 $\frac{\partial^2 z}{\partial y\partial x}$.

解 输入程序

```
>> syms x y z,F =2 * x +y + z - exp( -x -3 * y -2 * z);
Fy =diff(F,y); Fy =diff(F,z);
Zy = -diff(F,y)/diff(F,z),Zyx =diff(Zy,x)
```

运行后屏幕显示求 $\frac{\partial z}{\partial y}$ 和 $\frac{\partial^2 z}{\partial y\partial x}$ 的结果如下

```
Zy -
    ( -1 -3 * exp( -x -3 * y -2 * z))/(1 +2 * exp( -x -3 * y -2 * z))
Zyx -
    3 * exp( -x -3 * y -2 * z)/(1 +2 * exp( -x -3 * y -2 * z)) +2 * ( -1
-3 * exp( -x -3 * y -2 * z))/(1 +2 * exp( -x -3 * y -2 * z))^2 * exp( -x -3 * y
-2 * z)
```



习 题 8.1

1. 设三元函数 $u=3x^2+2\cos(5x^3-2y^5+21z^8)+\ln\frac{55xz-6xy-98z^4+356}{7x^{13}z+31y^{21}+9}$.

(1) 求 u 的一阶偏导函数和全微分；

(2) 求高阶偏导数 $\frac{\partial^3 u}{\partial x\partial y\partial z}$，$\frac{\partial^5 u}{\partial x^2\partial y\partial z^2}$ 和 $\frac{\partial^4 u}{\partial y^2\partial z\partial x}$；

(3) 求 z 在 $x=0,y=1,z=0$ 处的一阶偏导函数、全微分、高阶偏导数 $\left.\frac{\partial^3 u}{\partial x\partial y\partial z}\right|_{\substack{x=0\\y=1\\z=0}}$，$\left.\frac{\partial^5 u}{\partial x^2\partial y\partial z^2}\right|_{\substack{x=0\\y=1\\z=0}}$ 和 $\left.\frac{\partial^4 u}{\partial y^2\partial z\partial x}\right|_{\substack{x=0\\y=1\\z=0}}$.

2. 设一元函数 $y=3x^2+2\cos(5x^3-2)\sqrt{\ln(13x^5+6)}+\sqrt{\left(\frac{2x+21}{5x^2+6}\right)^3}$.

(1) 求 y 的五阶导数 $y^{(5)}(x)$ 和微分 dy；

(2) 求 y 在 $x=0$ 处的五阶导数 $y^{(5)}(0)$ 和微分 $dy\big|_{x=0}$.

3. 设二元函数 $z=\sqrt{\arctan(3x^2+61)}+2\sin(5x^3-2y^5)+\ln\frac{5x+6y+8}{7x^{13}+31y^{21}-9}$.

(1) 求 z 的一阶偏导函数和全微分；

(2) 求高阶偏导数 $\frac{\partial^2 z}{\partial x\partial y}$，$\frac{\partial^3 z}{\partial x\partial y^2}$，$\frac{\partial^2 z}{\partial y^2}$；

(3) 求 z 在 $x=0,y=1$ 处的一阶偏导函数、全微分、高阶偏导数 $\left.\frac{\partial^2 z}{\partial x\partial y}\right|_{\substack{x=0\\y=1}}$ 和 $\left.\frac{\partial^3 z}{\partial x^2\partial y}\right|_{\substack{x=0\\y=1}}$.

4. 由隐函数 $2x+y+z=e^{-x-3y-2z}$ 确定 $z=z(x,y)$,求 $\frac{\partial z}{\partial x}$,$\frac{\partial^2 z}{\partial x\partial y}$ 和 $\frac{\partial^3 z}{\partial x\partial y^2}$ 及 dz.

5. 设 $F(x+2y-z,x^2+y^2-z^2)=0$ 确定 $z=z(x,y)$,且 F 具有连续的偏导数,求 $\frac{\partial z}{\partial x}+\frac{\partial z}{\partial y}$.

6. 设 $\begin{cases} u^2+v^2-x^2-y=0, \\ -u+v-xy+1=0, \end{cases}$ 求 $\frac{\partial u}{\partial x}$ 和 $\frac{\partial y}{\partial u}$.

7. 设 $\begin{cases} x=-u^2+v, \\ y=u+v^2 \end{cases}$ 确定反函数组 $\begin{cases} u=u(x,y), \\ v=v(x,y), \end{cases}$ 求 $\frac{\partial u}{\partial x},\frac{\partial u}{\partial y},\frac{\partial v}{\partial x}$ 和 $\frac{\partial v}{\partial y}$.

8.2 一阶导数的数值计算及其 MATLAB 程序

导数的数值计算,或称数值微分,也称数值求导,是用离散方法近似地计算函数 $y=f(x)$ 在某点的导数值. 当然,通常只有函数以离散数值形式给出时才有必要用这种方法. 本节介绍一元函数的数值导数及其 MATLAB 程序.

8.2.1 差商求导及其 MATLAB 程序

根据导数定义

$$f'(x)=\lim_{h\to 0}\frac{\Delta y}{\Delta x}=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}, \tag{8.1}$$

Δx 和 Δy 分别称为自变量 x 和因变量 y 的增量,也称为差分. 可以用差分的商 $\frac{\Delta y}{\Delta x}$ 作微商(导数)的近似,从而有下面的求导数的近似值的定理.

定理 8.1(精度为 $O(h)$ 的前差公式和后差公式) 设函数 $y=f(x)$ 在 $[a,b]$ 上具有二阶连续导数,且 $x-h,x,x+h\in[a,b]$,则有

$$f'(x)\cong\frac{f(x+h)-f(x)}{h}, \tag{8.2}$$

$$f'(x)\cong\frac{f(x)-f(x-h)}{h}, \tag{8.3}$$

且存在 $\xi_i=\xi_i(x)\in(a,b)$ $(i=1,2)$,使得

$$f'(x)=\frac{f(x\pm h)-f(x)}{\pm h}+R_1(f,h), \tag{8.4}$$

其中 h $(h>0)$ 是绝对值很小的增量,估计导数 $f'(x)$ 的近似公式(8.2)和(8.3)式的截断误差公式为

$$R_1(f,h)=-\frac{h}{2}f''(\xi_i)=O(h)\ (i=1,2), \tag{8.5}$$

(8.2)和(8.3)式分别称为**前差公式和后差公式**.

由(8.1)式可得到(8.2)和(8.3)式. 为了估计这些近似公式的精度,将

$f(x+h)$和$f(x-h)$在点 x 处作泰勒展开

$$f(x \pm h)=f(x) \pm hf'(x)+\frac{h^2}{2}f''(\xi_i) \quad (i=1,2), \tag{8.6}$$

移项和化简可得到(8.4)和(8.5)式.

例 8.2.1 设$f(x)=\sin(5x^2-21)$.

(1) 分别利用前差公式和后差公式计算$f'(0.79)$的近似值和误差,取 4 位小数点计算,其中步长分别取 $h=0.1,0.01,0.001,0.0001$, $|f''(x)|\leqslant 80, x\in[0,1]$.

(2) 将(1) 中计算的$f'(0.79)$的近似值分别与精确值比较.

解 (1) 根据(8.2),(8.3)和(8.5)式编写计算 $y=f(x)$的一阶导数$f'(x)$的近似值和误差估计的 MATLAB 程序,并输入

```
>> x=0.79;h=[0.1,0.01,0.001,0.0001];
M=80;x1=x+h;x2=x-h; y=sin(5.*x.^2-21);
y1=sin(5.*x1.^2-21);y2=sin(5.*x2.^2-21);yq=(y1-y).\/h,yh=(y-y2).\/h,
wu=abs(h.*M/2),syms x,f=sin(5.*x.^2-21);dy=diff(f,x)
```

运行后屏幕显示利用前差公式和后差公式计算$f'(0.79)$的近似值 yq,yh 和误差估计 wu,取 4 位小数点计算,其中步长分别取 $h=0.1,0.01,0.001,0.0001$,$M=80$,导函数 dy.

```
yq =
1.46596380397978  4.22848550173043  4.44250759584697  4.46320955293622
yh =
5.96885352366536  4.68672022108227  4.48833808130555  4.46779260847907
wu =
4.00000000000000  0.40000000000000  0.04000000000000  0.00400000000000
dy =
10*cos(5*x^2-21)*x
```

(2) 计算$f'(0.79)$的值. 输入程序

```
>> x=0.79;dy =10*cos(5*x^2-21)*x,
wuq=abs(yq-dy), wuh=abs(yh-dy)
```

运行后屏幕显示利用前差公式和后差公式计算$f'(0.79)$的近似值与精确值的绝对误差 wuq,wuh 和$f'(0.79)$的精确值 dy 如下

```
dy =
    4.46550187104484
wuq =
2.99953806706506  0.23701636931441  0.02299427519787  0.00229231810861
wuh =
1.50335165262053  0.22121835003744  0.02283621026072  0.00229073743424
```

由(2)的计算结果可见,步长 h 越小,误差 wuh 越小,当 $h=0.0001$,利用前

差公式和后差公式计算$f'(0.79)$的近似值与精确值很接近,但是后差公式比前差公式计算的结果的误差小.

8.2.2 中心差商公式求导及其 MATLAB 程序

在这部分我们主要介绍两个问题,一是精度为$O(h^2)$的中心差商公式和误差估计,二是精度为$O(h^2)$的三点公式和误差估计及其 MATLAB 程序.

(一) 精度为$O(h^2)$的中心差商公式和误差估计

定理 8.2(精度为$O(h^2)$的中心差商公式) 设函数$y=f(x)$在$[a,b]$上具有三阶连续导数,且$x-h,x,x+h\in[a,b]$,则有

$$f'(x)\cong\frac{f(x+h)-f(x-h)}{2h},\tag{8.7}$$

且存在$\xi=\xi(x)\in(a,b)$,使得

$$f'(x)=\frac{f(x+h)-f(x-h)}{2h}+R_c(f,h),\tag{8.8}$$

其中$h\ (h>0)$是绝对值很小的增量,估计导数$f'(x)$的近似公式(8.7)的截断误差公式为

$$R_c(f,h)=-\frac{h^2}{6}f'''(\xi)=O(h^2),\quad \xi\in(a,b)\tag{8.9}$$

(8.7) 式的右式称**中心差商**,(8.7) 式称**中心差商公式**,它是最常用的公式.

证明 如果将(8.2)和(8.3)式平均一下,得(8.7)式. 为了估计导数的近似公式(8.7)的精度,将$f(x+h)$和$f(x-h)$在点x处作泰勒展开

$$f(x\pm h)=f(x)\pm hf'(x)+\frac{h^2}{2}f''(x)\pm\frac{h^3}{6}f'''(\xi_i)\ \ (i=1,2)\tag{8.10}$$

将(8.10)式中的两式相减,得

$$f(x+h)-f(x-h)=2hf'(x)+\frac{h^3}{6}[f'''(\xi_1)+f'''(\xi_2)].\tag{8.11}$$

因为函数$y=f(x)$在$[a,b]$上具有三阶连续导数,根据微分中值定理知,存在一点$\xi=\xi(x)\in(a,b)$,使得

$$f'''(\xi)=\frac{1}{2}[f'''(\xi_1)+f'''(\xi_2)]$$

代入(8.11)式后,移项整理得(8.8)式和(8.9)式.

让我们看一下(8.2),(8.3)和(8.7)式的几何意义. 在图 8-1 中$f'(x)$是切线AT的斜率,而(8.2),(8.3)和(8.7)式的右端分别是割线AB,AC和CB的斜率,显然CB的斜率更接近AT的斜率,即(8.7)式的精度更高. 不难看出(8.2)和(8.3) 式相当于用线性插值函数代替函数$f(x)$来求导数.

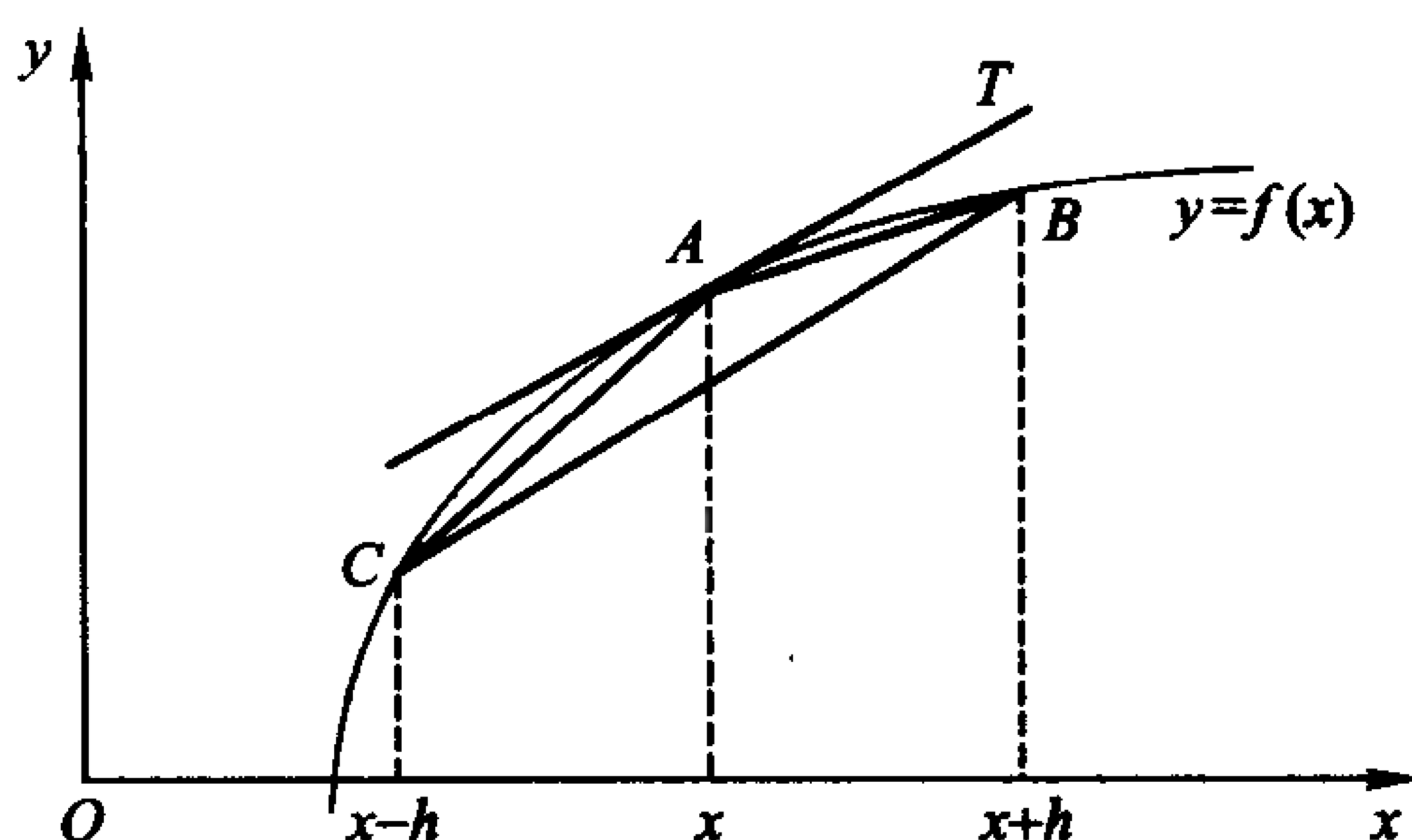


图 8-1 中心差商公式、前差公式和后差公式的几何意义

从以上的分析看，步长 h 越小，计算结果越精确，但这只是从截断误差角度而言的. 若从舍入误差角度看，h 很小时，$y(x_{k+1})$ 与 $y(x_{k-1})$ 很接近，二者相减会造成有效数字的严重损失，因此 h 不宜过小.

如果将数值求导区间 $[a,b]$ 进行 n 等分，步长 $h=(b-a)/n$，当函数 $y=f(x)$ 在分点上用离散数值表示为 (x_k,y_k)，$a=x_0<x_1<\cdots<x_n=b$ 时，函数在分点的导数值可以由(8.7)式得到

$$f'(x_k)\cong\frac{y(x_{k+1})-y(x_{k-1})}{2h},\quad k=1,2,\cdots n-1. \tag{8.12}$$

对于端点 x_0,x_n，用(8.2)和(8.3)式，有

$$f'(x_0)\cong\frac{y(x_1)-y(x_0)}{h},\quad f'(x_n)\cong\frac{y(x_n)-y(x_{n-1})}{h}, \tag{8.13}$$

误差为 $O(h)$.

(二) 精度为 $O(h^2)$ 的三点公式和误差估计及其 MATLAB 程序

为了提高精度用二次插值函数

$$P_2(x)=\frac{(x-x_{k+1})(x-x_{k+2})}{(x_k-x_{k+1})(x_k-x_{k+2})}y(x_k)+\frac{(x-x_k)(x-x_{k+2})}{(x_{k+1}-x_k)(x_{k+1}-x_{k+2})}y(x_{k+1})+$$
$$\frac{(x-x_k)(x-x_{k+1})}{(x_{k+2}-x_k)(x_{k+2}-x_{k+1})}y(x_{k+2})$$

代替曲线 $f(x)$，则

$$f'(x)\approx P_2'(x)$$
$$=\frac{(x-x_{k+1})+(x-x_{k+2})}{(x_k-x_{k+1})(x_k-x_{k+2})}y(x_k)+\frac{(x-x_k)+(x-x_{k+2})}{(x_{k+1}-x_k)(x_{k+1}-x_{k+2})}y(x_{k+1})+$$
$$\frac{(x-x_k)+(x-x_{k+1})}{(x_{k+2}-x_k)(x_{k+2}-x_{k+1})}y(x_{k+2}),$$

即

$$f'(x) \approx P_2'(x) = \frac{(x-x_{k+1})+(x-x_{k+2})}{2h^2}y(x_k) + \frac{(x-x_k)+(x-x_{k+2})}{-h^2}y(x_{k+1}) + \frac{(x-x_k)+(x-x_{k+1})}{2h^2}y(x_{k+2}). \tag{8.14}$$

根据插值余项公式(第六章定理6.1)知,存在 $\xi \in (\min(x_k,x_{k+1},x_{k+2}), \max(x_k,x_{k+1},x_{k+2}))$使得求导公式(8.14)的余项公式为

$$R_2(x) = f'(x) - P_2'(x) = \frac{f'''(\xi)}{6}((x-x_k)(x-x_{k+1})(x-x_{k+2}))' + \frac{1}{6}(x-x_k)(x-x_{k+1})(x-x_{k+2})\frac{\mathrm{d}}{\mathrm{d}x}f'''(\xi).$$

根据求导公式(8.14)及其余项公式,可以得到在 x_0,x_k,x_n处求导公式

$$f'(x_0) \cong P'(x_0) = \frac{-3y(x_0)+4y(x_1)-y(x_2)}{2h}, \tag{8.15}$$

$$f'(x_k) \cong P'(x_k) = \frac{f(x_k+h)-f(x_k-h)}{2h}\ (k=1,2,\cdots,n-1), \tag{8.16}$$

$$f'(x_n) \cong P'(x_n) = \frac{y(x_{n-2})-4y(x_{n-1})+3y(x_n)}{2h}, \tag{8.17}$$

及其余项公式

$$f'(x_0) - P'(x_0) = \frac{h^2}{3}f'''(\xi), \tag{8.18}$$

$$f'(x_k) - P'(x_k) = -\frac{h^2}{6}f'''(\xi)\ (k=1,2,\cdots,n-1), \tag{8.19}$$

$$f'(x_n) - P'(x_n) = \frac{h^2}{3}f'''(\xi), \tag{8.20}$$

其中 $\xi \in (x_0,x_n)$,其精度均为 $O(h^2)$. 其中(8.16)式实际上是(8.7)式,(8.15),(8.16)和(8.17)式统称**三点公式**.

根据精度为 $O(h^2)$的三点公式和误差估计公式编写计算 $f'(x)$的近似值和误差估计的 MATLAB 主程序如下:

利用精度为 $O(h^2)$的三点公式计算 $f'(x)$的近似值和误差估计的 MATLAB 主程序

输入的量:导数点组成的向量 $\boldsymbol{x}_i=(x_0,x_1,\cdots,x_n)$, 其中 $x_0,x_1,\cdots,x_n$成等差数列,公差为 h,函数 $y=f(x)$在 $x=x_0,x_1,\cdots,x_n$处的函数值组成的向量 $\boldsymbol{f}_i$, $M=\max\limits_{x_0<\xi<x_n}\{|f'''(\xi)|\}$;

输出的量:向量 $\boldsymbol{x}=(x_0,x_1,\cdots,x_n)$, yx 是函数 $y=f(x)$在 $x=x_0,x_1,\cdots,x_n$处的导数的近似值, wuc 是误差估计.

```
function[n,xi,yx,wuc] = sandian(h,xi,fi,M)
  n = length(fi); yx = zeros(1,n); wuc = zeros(1,n);
  x1 = xi(1);x2 = xi(2);x3 = xi(3);
  y1 = fi(1);y2 = fi(2);y3 = fi(3);xn = xi(n);xn1 = xi(n-1);
  xn2 = xi(n-2);yn = fi(n);yn1 = fi(n-1);yn2 = fi(n-2);
for k = 2:n-1
  yx(1) = (-3*y1+4*y2-y3)/(2*h);yx(n) = (yn2-4*yn1+3*
yn)/(2*h);
  yx(2) =(fi(3)-fi(1))/(2*h);yx(k) =(fi(k+1)-fi(k-1))./(2*h);
  wuc(1) = abs(h.^2.*M./3);wuc(n) = abs(h.^2.*M./3);
  wuc(2:n-1) = abs(-h.^2.*M./6);
end
```

例 8.2.2(人口增长率)　已知 20 世纪美国人口统计数据如表 8－3,试计算表中这些年份的人口增长率.

表 8－3　20 世纪美国人口统计数据

年份	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990
人口($\times 10^6$)	76.0	92.0	106.5	123.2	131.7	150.7	178.3	204.0	226.5	251.4

又已知某地区 20 世纪 70 年代的人口增长率如表 8－4,且 1970 年人口为 210(百万),试估计 1980 年的人口.

表 8－4　某地区 20 世纪 70 年代人口增长率数据

年份	1970	1972	1974	1976	1978	1980
年增长率(%)	0.87	0.85	0.89	0.91	0.95	1.10

解　(1) 若记时刻 t 的人口为 $y(t)$,则人口(相对)增长率为 $r(t)=\dfrac{\frac{dy}{dt}}{y(t)}$,表示每年人口增长的比例. 对于题目给出的人口数据(表 8－5),记 $x_0=1900$ 年,$h=10$,相应地,人口记为 y_k,年增长率为 r_k,则 $x_1=1910, x_2=1920, \cdots, x_9=1990$ 年. 用三点公式(8.15),(8.16) 和(8.17)得

$$r_k=\frac{y_{k+1}-y_{k-1}}{20y_k},\quad k=1,2,\cdots,8,$$

$$r_0=\frac{-3y_0+4y_1-y_2}{20y_0},\quad r_9=\frac{y_7-4y_8+3y_9}{20y_9}.$$

输入程序

```
>> h=10;xi =1900:10:1990;M=1;
fi=[76.0,92.0,106.5,123.2,131.7,150.7,178.3,204.0,226.5,251.4];
[n,xi,yx,wuc]=sandian(h,xi,fi,M), rk=(yx./fi)*100
```

运行后,整理的计算结果为(将每10年的增长率变为每年的增长率):

年份	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990
r_k(%)	2.20	1.66	1.46	1.02	1.04	1.55	1.49	1.18	1.05	1.04

可以看出,20世纪美国人口增长率总的来说在下降,但是有起伏,20世纪30年代和第二次世界大战时期人口增长率显著下降,战后又迅速上升,八九十年代稳定在1%多一点.

(2) 仍用上面的记号,人口增长满足微分方程$\frac{dy}{dt}=r(t)\cdot y(t)$,它在初始条件$y(0)=y_0$下的解为

$$y(t)=y_0 e^{\int_0^t r(u)du}.$$

因为在题目中增长率$r(u)$以离散数据给出(表8-6),所以上式要用数值积分计算.用梯形公式计算的结果是1980年该地区人口为230.2(百万).

当$n=2$时,节点$x_0,x_1=x_0+h,x_2=x_0+2h$,则在$x=x_0+th$处的导数公式为

$$P'(x_0+th)=\frac{1}{2h}[(2t-3)y(x_0)-4(t-1)y(x_1)+(2t-1)y(x_2)].$$

当$t=0,1,2$时,上式便是在x_0,x_1,x_2处的三点公式.根据这个公式及其误差公式(8.18)、(8.19)和(8.20)式,取$M=\max\limits_{x_0<\xi<x_n}\{|f'''(\xi)|\}$,编写了下面名为sandian3.m的计算函数$f(x)$在$x=x_0,x_1,x_2$处的导数的近似值的MATLAB主程序.

利用精度为$O(h^2)$的三点公式计算$f'(x)$的近似值和误差估计的MATLAB主程序

输入的量:导数点组成的向量$\boldsymbol{x}_i=(x_0,x_1,x_2)$,其中$x_0,x_1,x_2$成等差数列,公差为$h$,函数$y=f(x)$在$x=x_0,x_1,x_2$处的函数值组成的向量$\boldsymbol{f}_i$,$M=\max\limits_{x_0<\xi<x_n}\{|f'''(\xi)|\}$;

输出的量:向量$\boldsymbol{x}=(x_0,x_1,x_2)$,$yx$是函数$y=f(x)$在$x=x_0,x_1,x_2$处的导数的近似值,$wuc$是误差估计.

```
function [x,yxj,wuc] = sandian3(h,xi,fi,M)
  yxj = zeros(1,3);wuc = zeros(1,3);x1 = xi(1);
  x2 = xi(2);x3 = xi(3);y1 = fi(1);y2 = fi(2);y3 = fi(3);
for t = 1:3
  s(t) = ((2*t-5)*y1-4*(t-2)*y2+(2*t-3)*y3)/(2*h);x = xi;y = s(t);
yxj(t) = y;
  if t = = 2
```

```
    wuc(2)=abs(-h.^2*M/6);
  else
    wuc(1:2:3)=abs(h.^2*M/3);
  end
end
```

例 8.2.3 设已给出 $y=f(x)$ 的数据表 8-5:

表 8-5

x	1.000 0	1.100 0	1.200 0	1.300 0	1.400 0	1.500 0	1.600 0
$f(x)$	0.250 0	0.226 8	0.206 6	0.189 0	0.173 6	0.160 0	0.147 9

$M=0.7502$,试用三点公式计算下列问题:

(1) 当 $h=0.1$ 时,$y=f(x)$ 在 $x=1.0000,1.1000,1.2000,1.3000,1.4000,1.5000,1.6000$ 处的一阶导数的近似值,并估计误差;

(2) 当 $h=0.2$ 时,$y=f(x)$ 在 $x=1.0000,1.2000,1.4000,1.6000$ 处的一阶导数的近似值,并估计误差;

(3) 当 $h=0.3$ 时,$y=f(x)$ 在 $x=1.0000,1.3000,1.6000$ 处的一阶导数的近似值,并估计误差;

(4) 表 8-5 中的数据是函数 $f(x)=\frac{1}{(1+x)^2}$ 在相应点的数值,将(1)至(3)计算的一阶导数的近似值与 $f(x)=\frac{1}{(1+x)^2}$ 的一阶导数值比较,并求出它们的绝对误差.

解 (1) 保存 M 文件 sandian.m,sandian3.m;

(2) 在 MATLAB 工作窗口输入如下程序

```
>> syms x,y=1/((1+x)^2);yx=diff(y,x,1),
yx3=diff(y,x,3),
```

运行后将屏幕显示的结果为

```
yx =                    yx3 =
   -2/(1+x)^3              -24/(1+x)^5
```

(3) 在 MATLAB 工作窗口输入如下程序

```
>>h=0.1; xi=1.0000:h:1.6000;
fi=[0.2500 0.2268 0.2066 0.1890 0.1736 0.1600 0.1479];
x=1:0.001:1.6; yx3 = -24./(1+x).^5;M=max(abs(yx3));
[n1,x1,yx1,wuc1]=sandian(h,xi,fi,M)
yxj1 = -2./(1+xi).^3,wuyxj1=abs(yxj1-yx1)
h=0.2;xi=1.0000:h:1.6000;fi=[0.2500  0.2066  0.1736  0.1479];
x=1:0.001:1.6;yx3 = -24./(1+x).^5;M= max(abs(yx3));
```

```
    [n2,x2,yx2,wuc2] = sandian(h,xi,fi,M)
    yxj2 = -2./(1 +xi).^3,wuyxj2 = abs((yxj2 - yx2))
    h =0.3;xi =1.0000:h:1.6000;fi =[0.2500 0.1890 0.1479];
    x =1:0.001:1.6;yx3  = -24./(1 +x).^5;M = max(abs(yx3));
    [x3,yx3, wuc3] = sandian3(h,xi,fi,M)
    yxj3 = -2./(1 +xi).^3,wuyxj3 = abs(yxj3 -yx3)
或 >> h1 =0.1,
    x =[1.0000,1.1000,1.2000,1.3000,1.4000,1.5000,1.6000];
    f =[0.2500,0.2268,0.2066,0.1890,0.1736,0.1600,0.1479];
    xi =x(1:3);f11 = f(1:3);M = 0.7502;
    [x11,yxj11,wuc11] = sandian3(h1,xi,f11,M),
    xi = x(4:6);f12 = f(4:6);
    [x12,yxj12,wuc12] = sandian3(h1,xi,f12,M),
    xi =x(5:7);f13 = f(5:7);
    [x13,yxj13,wuc13] = sandian3(h1,xi,f13,M),
    h2 =0.2,xi = x(1:2:5);f21 = f(1:2:5);
    [x21,yxj21,wuc21] = sandian3(h2,xi,f21,M),
    xi =x(2:2:6);f22 = f(2:2:6);
    [x22,yxj22,wuc22] = sandian3(h2,xi,f22,M),
    xi =x(3:2:7);f23 = f(3:2:7);
    [x23,yxj23,wuc23] = sandian3(h2,xi,f23,M),
    h3 =0.3,xi = x(1:3:7);f31 = f(1:3:7);
    [x31,yxj31,wuc31] = sandian3(h3,xi,f31,M),
```

将运行的结果列入表 8-6 中. 结果表明:三点公式比前差公式和后差公式计算准确,步长越小,计算越准确,这是在三阶导数存在和舍入误差不超过截断误差的前提下.

表 8-6 例 8.2.3 计算结果

x	1.000 0	1.100 0	1.200 0	1.300 0	1.400 0	1.500 0	1.600 0
$f'(x)$精确值	-0.250 0	-0.216 0	-0.187 8	-0.164 4	-0.144 7	-0.128 0	-0.113 8
$h=0.1, f'(x)$近似值	-0.247 0	-0.217 0	-0.189 0	-0.165 0	-0.145 0	-0.128 5	-0.113 5
$h=0.2, f'(x)$近似值	-0.243 0		-0.191 0		-0.146 7		-0.110 3
$h=0.3, f'(x)$近似值	-0.236 5			-0.170 2			-0.103 8
$h=0.1$,绝对误差	0.003 0	0.001 0	0.001 2	0.000 6	0.000 3	0.000 5	0.000 3
$h=0.1$,误差估计	0.002 5	0.001 3	0.001 3	0.001 3	0.001 3	0.001 3	0.002 5
$h=0.2$,绝对误差	0.007 0		0.003 2		0.002 1		0.003 5
$h=0.2$,误差估计	0.010 0		0.005 0		0.005 0		0.010 0
$h=0.3$,绝对误差	0.013 5			0.005 8			0.010 0
$h=0.3$,误差估计	0.022 5			0.011 3			0.022 5

8.2.3 理查森(Richardson)外推法求导及其 MATLAB 程序

从求解$f'(x)$的低阶公式中推导出高阶公式的方法称为理查森**外推法**. 外推法可以提高计算精度. 在这部分我们主要介绍三个问题,一是一般形式的理查森外推法及其 MATLAB 程序,二是精度为$O(h^4)$的中心差商公式和误差估计及其 MATLAB 程序,三是变步长的中心差商公式及其 MATLAB 程序.

(一) 一般形式的理查森外推法及其 MATLAB 程序

如果三阶导数$f'''(\xi)$的值变化不大,则导数$f'(x)$的近似公式(8.7)的截断误差(8.9)以与h^2同样的速度趋近于零,即$O(h^2)$. 当计算机进行计算时,不宜将h取得太小. 因此,如果求解导数$f'(x)$的近似值公式具有精度为$O(h^4)$或更高精度的截断误差,则对计算机计算问题是很有用的. 下面的定理描述了提高计算机精度的一般形式.

定理 8.3(理查森外推法) 设函数$y=f(x)$在x_0处的导数$y'=f'(x_0)$具有精度为$O(h^{2k})$的两个近似值$D_{k-1}(h)$和$D_{k-1}(2h)$,而且满足

$$f'(x_0)=D_{k-1}(h)+\xi_1h^{2k}+\xi_2h^{2k+2}+\cdots, \tag{8.21}$$

和

$$f'(x_0)=D_{k-1}(2h)+4^k\xi_1h^{2k}+4^{k+1}\xi_2h^{2k+2}+\cdots. \tag{8.22}$$

则有改进的$f'(x_0)$近似值表达式

$$f'(x_0)\approx D_k(h)=\frac{4^kD_{k-1}(h)-D_{k-1}(2h)}{4^k-1}, \tag{8.23}$$

满足

$$f'(x_0)=D_k(h)+O(h^{2k+2}).$$

我们称(8.23)式为**理查森外推公式**,用(8.23)式计算$f'(x_0)$近似值的方法称为**理查森外推法**.

证明 用4^k乘以(8.21)式的两边,再减去(8.22)式,得

$$(4^k-1)f'(x_0)=4^kD_{k-1}(h)-D_{k-1}(2h)+4^kh^{2k+2}(-3\xi_2+\cdots)$$

上式两边同除以(4^k-1),得

$$f'(x_0)=\frac{4^kD_{k-1}(h)-D_{k-1}(2h)}{4^k-1}+O(h^{2k+2}).$$

例 8.2.4 试利用理查森外推法,由精度为$O(h^2)$的中心差商公式推导出精度为$O(h^4)$的差商公式.

解 因为精度为$O(h^2)$,步长为h的中心差商公式为

$$D_{k-1}(h)=\frac{f(x_0+h)-f(x_0-h)}{2h},$$

且
$$f'(x_0)=D_{k-1}(h)+O(h^2).$$
则步长为 $2h$ 的中心差商公式为
$$D_{k-1}(2h)=\frac{f(x_0+2h)-f(x_0-2h)}{4h},$$
且
$$f'(x_0)=D_{k-1}(2h)+O(h^2).$$

由定理 8.3，得
$$f'(x_0)=\frac{4D_{k-1}(h)-D_{k-1}(2h)}{4-1}+O(h^{2+2}).$$
而
$$\frac{4D_{k-1}(h)-D_{k-1}(2h)}{4-1}=\frac{4}{3}\frac{f(x_0+h)-f(x_0-h)}{2h}-\frac{1}{3}\cdot\frac{f(x_0+2h)-f(x_0-2h)}{4h}$$
$$=\frac{8f(x_0+h)-8f(x_0-h)-f(x_0+2h)+f(x_0-2h)}{12h},$$
$$f'(x_0)=\frac{8f(x_0+h)-8f(x_0-h)-f(x_0+2h)+f(x_0-2h)}{12h}+O(h^4),$$
$$f'(x)\cong\frac{8f(x+h)-8f(x-h)-f(x+2h)+f(x-2h)}{12h}. \tag{8.24}$$
因此，利用理查森外推法，可以由精度为 $O(h^2)$ 的中心差商公式推导出精度为 $O(h^4)$ 的差商公式(8.24). 公式 (8.24) 称为**精度为 $O(h^4)$ 的中心差商公式**.

我们可以根据(8.23)式编写 MATLAB 程序计算 $y=f(x)$ 的导数 $y'=f'(x)$ 的近似值的 M 函数文件，保存起来，计算时直接调用. 在这个程序中，构造近似值 $D(j,k)$，$k\leqslant j$ 的表，并将 $f'(x)\approx D(n,n)$ 作为最终答案. 近似值 $D(j,k)$ 存放在下三角形矩阵中. 第 j 行的元用与(8.23)式等价的表达式
$$D(j,k)=D(j,k-1)+\frac{D(j,k-1)-D(j-1,k-1)}{4^k-1},\ 1\leqslant k\leqslant j,$$
第一列为
$$D(j,0)=\frac{f(x+2^{-j}h)-f(x-2^{-j}h)}{2^{-j+1}h}.$$

利用理查森外推法计算 $f'(x)$ 的近似值和误差估计的 MATLAB 程序

输入的量：*fun* 是函数 $y=f(x)$，x_0 是导数点，*jdwc* 是 $f'(x)$ 的近似值的绝对误差，*max*1 是计算次数的最大值.

输出的量：dy 是导数 $f'(x_0)$ 近似值，**Dy** 是导数 $f'(x_0)$ 近似值的迭代矩阵，*jdW* = |**Dy**(n,n) − **Dy**$(n,n-1)$| 的值，n 是最佳近似值 dy 的坐标.

```
function [Dy,dy,jdw,n] = diffext1( fun,x0,jdwc,max1)
h =1;j =1; n =1;jdW =1;xdW =1; x1 =x0 +h;x2 =x0 -h;
Dy(1,1) =(feval(fun,x1) - feval(fun,x2))/(2 * h);
```

```
while((jdW > jdwc)&(j < max1))
  j;x1 = x0 +2^( - j) * h;x2 = x0 - 2^( - j) * h;
  Dy(j +1,1) = (feval(fun,x1) - feval(fun,x2))/(2^(1 - j) * h);
  for k = 1:j
  k;Dy(j +1,k +1) = Dy(j +1,k) + ( Dy(j +1,k) - Dy(j,k))/(4^k -1);
  end
  jdW = abs( Dy( j +1,j +1) - Dy( j +1,j));j = j +1;
end
[n,n] = size( Dy);jdw = abs( Dy( n,n) - Dy( n,n -1));
dy = Dy( n,n);
```

例 8.2.5 设 $f(x)=\sin(5x^2-21)$，取精度 $jdwc=0.0000001$，用理查森外推法计算 $f'(0.79)$ 的近似值和误差估计，将近似值与精确值 4.46550187104484 比较.

解 (1) 保存以 fun.m 命名的 M 函数文件

```
function y = fun( x)
y = sin( 5 .* x.^2 -21);
```

取最大计算次数 max1 =100，输入程序

```
>> x0 =0.79;jdwc =0.0000001,max1 =100;
[Dy,dy,jdw,n] = diffext1(@ fun,x0,jdwc,max1),
wu =4.46550187104484 - dy
```

运行后屏幕显示 $f'(0.79)$ 的近似值 dy，dy 与精确值的绝对误差 wu，导数 $f'(x_0)$ 近似值的迭代矩阵 $\boldsymbol{Dy}$，$jdW=|\boldsymbol{Dy}(n,n)-\boldsymbol{Dy}(n,n-1)|$ 的值，最佳近似值 dy 的坐标 n 如下

```
Dy =
   Columns 1 through 4
0.95036708207779                  0                  0                  0
0.87447447334140   0.84917693709594                  0                  0
1.04543344913993   1.10241977440611   1.11930263022679                  0
3.33291848491782   4.09541349684379   4.29494641167297   4.34535345582291
4.16327772677060   4.44006414072153   4.46304085031338   4.46570901600608
4.38872901334021   4.46387944219674   4.46546712896176   4.46550564132126
4.44623223629059   4.46539997727405   4.46550134627921   4.46550188941123
  Columns 5 through 7
                 0                  0                  0
                 0                  0                  0
                 0                  0                  0
                 0                  0                  0
  4.46618099859504                  0                  0
```

```
  4.46550484377347   4.46550418282057                   0
  4.46550187469786   4.46550187179553   4.46550187123118
      dy =               jdw =                  n =       wu =
4.46550187123118  5.643530087695581e-010  7  -1.863398324530863e-010
```

（二）精度为 $O(h^4)$ 的中心差商公式和误差估计及其 MATLAB 程序

我们将例 8.2.4 中利用理查森外推法证明的结论归纳为下面的定理.

定理 8.4（精度为 $O(h^4)$ 的中心差商公式） 设函数 $y=f(x)$ 在 $[a,b]$ 上具有五阶连续导数，且 $x\pm h, x, x\pm 2h\in[a,b]$，则有

$$f'(x)\cong\frac{8f(x+h)-8f(x-h)-f(x+2h)+f(x-2h)}{12h},$$

且存在 $\xi=\xi(x)\in(x_0-2h,x_0+2h)$，使得

$$f'(x)=\frac{8f(x+h)-8f(x-h)-f(x+2h)+f(x-2h)}{12h}+R_{c4}(f,h),\quad(8.25)$$

其中 h $(h>0)$ 为很小的增量，且估计导数 $f'(x)$ 的近似公式(8.24)的截断误差公式为

$$R_{c4}(f,h)=\frac{h^4}{30}f^{(5)}(\xi)=O(h^4).\qquad(8.26)$$

根据精度为 $O(h^4)$ 的中心差商公式(8.24)，(8.25) 和(8.26)式编写了下面名为 zxcs4.m 和 zxcs5.m 的计算函数 $f(x)$ 在 $x=x\pm h, x, x\pm 2h\in[a,b]$ 处的导数的近似值的 MATLAB 主程序.

> **用精度为 $O(h^4)$ 的中心差商公式计算 $f'(x)$ 的近似值和误差估计的 MATLAB 程序**
>
> 输入的量：x_0 是导数点，h 是步长，f_i 是函数 $y=f(x)$ 在 $x_i=x_0\pm h, x_0, x_0\pm 2h$ 处的函数值组成的向量，$M=\max\limits_{x-2h<\xi<x+2h}\{|f^{(5)}(\xi)|\}$；
>
> 输出的量：yx 是函数 $y=f(x)$ 在 x_0 处的导数的近似值，wuc 是误差估计.

```
function[x0,yx,wuc]=zxcs4(h,x0,fi,M)
xi=[x0-2*h,x0-h,x0,x0+h,x0+2*h];
x1=xi(1);x2= xi(2);x3= xi(3);x4= xi(4);
x5=xi(5);y1=fi(1);y2=fi(2);y3=fi(3);
y4=fi(4);y5=fi(5);
yx=(8*y4-8*y2-y5+y1)/(12*h);
wuc=abs(h.^4*M/30);
```

如果给出函数，则用下面的程序：

用精度为 $O(h^4)$ 的中心差商公式计算 $f'(x)$ 的近似值和误差估计的 MATLAB 程序

输入的量：*fun* 是函数 $y=f(x)$，x_0 是导数点，h 是步长，$M=\max\limits_{x-2h<\xi<x+2h}\{|f^{(5)}(\xi)|\}$.

输出的量：$\boldsymbol{x}=(x_0-2h,x_0-h,x_0,x_0+h,x_0+2h)$，$\boldsymbol{y}$ 是 $y=f(x)$ 在 $\boldsymbol{x}$ 处函数值向量，yx 是函数 $y=f(x)$ 在 x_0 处的导数的近似值，wuc 是误差估计.

```
function[x,y,yx,wuc] = zxcs5(fun,h,x0, M)
x = zeros(1,5);y = zeros(1,5);
for k =1:5
    x(k) = x0 + (k -3).* h;
    y(k) = feval(fun,x(k));
end
x;y;
yx = (8 * y(4) -8 * y(2) -y(5) +y(1))/(12 * h);
wuc = abs(h.^4 * M/30);
```

例 8.2.6 设 $f(x)=\sin(5x^2-21)$.

(1) 分别根据(8.7)式和(8.24)式计算 $f'(0.79)$ 的近似值，并估计误差，取小数点后 14 位计算，其中步长分别取 $h=0.1, 0.01, 0.001, 0.0001$，$|f'''(x)|\leqslant M=872, x\in[0,1]$.

(2) 将(1) 中计算的 $f'(0.79)$ 的近似值分别与精确值比较.

解 (1) 计算 $y=f(x)$ 的一阶导数 $f'(x)$ 的近似值和误差估计.

方法 1 保存以 fun.m 命名的 M 函数文件

```
function y  = fun(x)
y = sin(5.* x.^2 -21);
```

输入程序

```
>> x0 =0.79;h1 =0.1,M =872;[x,y,yx1,wuc1] = zxcs5(@ fun,h1,x0,M)
h2 =0.01,[x,y,yx2,wuc2] = zxcs5(@ fun,h2,x0,M),
h3 =0.001,[ x,y,yx3,wuc3] = zxcs5(@ fun,h3,x0, M),
h4 =0.0001,[x,y,yx4,wuc4] = zxcs5(@ fun,h4,x0, M),
```

运行后屏幕显示 $\boldsymbol{x}=(x_0-2h,x_0-h,x_0,x_0+h,x_0+2h)$，$y=f(x)$ 在 $\boldsymbol{x}$ 处函数值向量 $\boldsymbol{y}$，步长分别取 $h=0.1,0.01,0.001,0.0001$ 时，根据(8.24)式计算函数 $y=f(x)$ 在 x_0 处的导数的近似值 yx_1,yx_2,yx_3,yx_4 和误差估计 wuc_1,wuc_2,wuc_3,wuc_4 如下

```
x =
  Columns 1 through 3
 0.59000000000000   0.69000000000000   0.79000000000000
```

```
  Columns 4 through 5
 0.89000000000000   0.99000000000000
y =
  Columns 1 through 3
 -0.39855804055354   0.22803197210174   0.82491732446828
  Columns 4 through Column 5
 0.97151370486625   0.38160930304430
yx1  =                       wuc1 =
     4.30640543209856             0.00290666666667
yx2  =                       wuc2 =
     4.46548476000396             2.906666666666667e-007
yx3  =                       wuc3  =
     4.46550186933213             2.906666666666667e-011
yx4  =                       wuc4  =
     4.46550187103480             2.906666666666667e-015
```

同理,可以求出根据(8.7)式计算$f'(0.79)$的近似值及其误差估计值.

方法2 根据定理8.2和定理8.4编写计算$y=f(x)$的一阶导数$f'(x)$的近似值和误差估计的MATLAB程序,并输入此程序

```
>> x=0.79;h=[0.1,0.01,0.001,0.0001];M=872;x1=x+h;x2=x-h;
x3=x+2*h;x4=x-2*h;y1=sin(5.*x1.^2-21);
y2=sin(5.*x2.^2-21);
y3=sin(5.*x3.^2-21);y4=sin(5.*x4.^2-21);
yc2=(y1-y2)./(2*h),
wuc2=abs(h.^2*M/6),yc4=(8*y1-8*y2-y3+y4)./(12*h),
wuc4=abs(h.^4*M/30),syms x,f=sin(5.*x.^2-21);dy=diff(f,x)
```

运行后屏幕显示步长分别取$h=0.1,0.01,0.001,0.0001$,$M=872$,分别根据(8.7)式和(8.24)式计算$f'(0.79)$的近似值yc_2,yc_4及其误差估计值wuc_2,wuc_4,导函数dy如下

```
      yc2  =
3.71740866382257  4.45760286140635  4.46542283857626  4.46550108070765
      wuc2  =
1.45333333333333  0.01453333333333  0.00014533333333  0.00000145333333
      yc4  =
4.30640543209856  4.46548476000396  4.46550186933213  4.46550187103480
      wuc4  =
0.00290666666667  0.00000029066667  0.00000000002907  0.00000000000000
      dy  =
      10*cos(5*x^2-21)*x
```

(2) 计算 $f'(0.79)$ 的值. 输入程序

```
>> x =0.79;dy  =10 * cos(5 * x^2 -21) * x,
wu2 = abs( yc2 - dy),wu4 = abs( yc4 - dy)
```

运行后屏幕显示 $f'(0.79)$ 的近似值与精确值的绝对误差 wuc_2, wuc_4 和 $f'(0.79)$ 的精确值 dy 如下

```
dy =
    4.46550187104484
wuc2 =
0.74809320722227  0.00789900963848  0.00007903246857  0.00000079033719
wuc4 =
0.15909643894627  0.00001711104088  0.00000000171271  0.00000000001003
```

由(2) 计算的结果可见,步长越小,误差越小,且精度为 $O(h^4)$ 的中心差商公式比精度为 $O(h^2)$ 的中心差商公式计算近似值的精度高,当 $h=0.0001$,利用精度为 $O(h^4)$ 的中心差商公式计算 $f'(0.79)$ 的近似值与精确值最为接近,误差仅为 $wuc_4 = 1.003\times10^{-11}$.

(三) 变步长的中心差商公式及其 MATLAB 程序

从截断误差角度而言,步长 h 越小,计算结果越精确,但是从舍入误差角度看,h 很小时,$y(x_{k+1})$ 与 $y(x_{k-1})$ 很接近,二者直接相减会造成有效数字的严重损失,所以 h 不宜过小. 因此,在实际计算时,人们希望在保证截断误差满足精度要求的前提下选择尽可能大的步长. 然而事先给出一个合适的步长往往是困难的. 通常在变步长的过程中实现步长的自动选取. 一种自动选取步长的方法是,将精度为 $O(h^2)$ 的中心差商公式(8.7),即

$$f'(x) \cong \frac{f(x+h)-f(x-h)}{2h}$$

中的步长 h 用 $h/10^k$ 代换,得到变步长的中心差商公式序列

$$y' = f'(x) \approx D_k = \frac{f(x+h/10^k)-f(x-h/10^k)}{2h/10^k},\ k=0,1,2,\cdots,n. \quad (8.27)$$

计算停止条件是 $|D_{n+1}-D_n| \geqslant |D_n - D_{n-1}|$ 或 $|D_n - D_{n-1}| < \varepsilon$ (其中 ε 是事先给定的误差). 可以用 $|D_n - D_{n-1}| < \varepsilon$ 求最佳近似值 $y' = f'(x) \approx D_n$. 根据(8.27)式编写了用变步长的中心差商公式计算 $f'(x)$ 的近似值和误差估计的 MATLAB 程序如下:

用变步长的中心差商公式计算 $f'(x)$ 的近似值和误差估计的 MATLAB 程序

输入的量:fun 是函数 $y=f(x)$,x_0 是导数点,wu 是 $f'(x)$ 的近似值的精度,$max1$ 是计算次数的最大值,h_0 是初始步长.

输出的量:H 表示步长 $h/10^k$ 组成的数组,Dy 是导数值组成的数组,W 是误差限数组,n 是迭代次数.

```
function[n,H,Dy,W] = difflim(fun,x0,h0,wu,max1)
Dy = zeros(1,max1);W = zeros(1,max1);H = zeros(1,max1);
h = h0;H(1) = h;E(1) = 0;x1 = x0 + h;x2 = x0 - h;
h1 = h/10;x3 = x0 + h1;x4 = x0 - h1;
Dy(1) = (feval(fun,x1) - feval(fun,x2))./(2 * h);
Dy(2) = (feval(fun,x3) - feval(fun,x4))./(2 * h1);
W(1) = abs(Dy(2) - Dy(1));
k = 1;
while((W(k) > wu)&(k < max1))
h = h/(10^(k));H(k + 1) = h;x1 = x0 + H(k + 1);x2 = x0 - H(k + 1);
Dy(k + 1) = (feval(fun,x1) - feval(fun;x2))./(2 * H(k + 1));
W(k + 1) = abs(Dy(k + 1) - Dy(k));
k = k + 1;
end
n = length(Dy(1:k)) - 2;
Dy = Dy(2:k);W = W(2:k);H = H(2:k);
```

例 8.2.7 设 $f(x) = \sin(5x^2 - 21)$. 取初始步长 $h_0 = 0.2$, 精度 $wu = 0.000\,01$, 用变步长的中心差商公式计算 $f'(0.79)$ 的近似值和误差估计, 与精确值 4.465 501 871 044 84 比较.

解 保存以 fun.m 命名的 M 函数文件

```
function y = fun(x)
y = sin(5.*x.^2 -21);
```

输入程序

```
>> x0 = 0.79;wu = 0.00001;max1 = 100;h0 = 0.2;
[n,H,Dy,W] = difflim(@fun,x0,h0,wu,max1)
jdwc = 4.46550187104484 - Dy
```

运行后屏幕显示迭代次数 n, 变步长数组 H, 用变步长的中心差商公式计算 $f'(0.79)$ 的近似值 Dy, 误差估计值 W, $jdwc$, 精确值与近似值 Dy 的差如下

```
n =
    2
H =
   0.02000000000000   0.00020000000000   0.00000020000000
Dy =
   4.43395716561354   4.46549870972618   4.46550187410688
W =
   2.48353880661895   0.03154154411264   0.00000316438070
jdwc =
   0.03154470543130   0.00000316131866  -0.00000000306204
```

8.2.4 牛顿多项式求导及其 MATLAB 程序

在 6.3 节牛顿插值中，如果函数 $f(t)$ 在 $[a,b]$ 上有定义，对于 $n+1$ 个节点 (t_j,y_j)，$j=0,1,\cdots,n$，其中 $t_j\in[a,b]$ 互不相同，满足 $f(t_j)=y_j$，$j=0,1,\cdots,n$. 对于任意 $t\in[a,b]$，则有过 $n+1$ 个节点的 $f(t)$ 的 n **阶牛顿插值多项式**

$$P_n(t)=f(t_0)+f[t_0,t_1](t-t_0)+f[t_0,t_1,t_2](t-t_0)(t-t_1)+\cdots+ f[t_0,t_1,t_2,\cdots,t_n](t-t_0)(t-t_1)\cdots(t-t_{n-1}) \tag{8.28}$$

和牛顿插值多项式的余项

$$R_n(t)=f[t,t_0,t_1,\cdots,t_n](t-t_0)(t-t_1)\cdots(t-t_{n-1})(t-t_n),$$

使得带余项的牛顿插值公式为

$$f(t)=P_n(t)+R_n(t),$$

其中一阶差商为

$$f[t,t_0]=\frac{f(t)-f(t_0)}{t-t_0},$$

有 $f(t)=f(t_0)+f[t,t_0](t-t_0)$.

类似地，由二阶差商至 n 阶差商的定义得到下列方程

$$f[t,t_0]=f[t_0,t_1]+f[t,t_0,t_1](t-t_1),$$

$$f[t,t_0,t_1]=f[t_0,t_1,t_2]+f[t,t_0,t_1,t_2](t-t_2),$$

$$\cdots\cdots$$

$$f[t,t_0,t_1,\cdots,t_{n-1}]=f[t_0,t_1,t_2,\cdots,t_n]+f[t,t_0,t_1,\cdots,t_n](t-t_n).$$

对牛顿插值多项式 $P_n(t)$ 求导，得

$$\begin{aligned}P'_n(t)=&f[t_0,t_1]+f[t_0,t_1,t_2][(t-t_0)+(t-t_1)]\\&+f[t_0,t_1,t_2,t_3][(t-t_0)(t-t_1)\\&+(t-t_0)(t-t_2)+(t-t_1)(t-t_2)]\\&+\cdots+f[t_0,t_1,t_2,\cdots,t_n]\sum_{k=0}^{n-1}\prod_{\substack{j=0\\j\neq k}}^{n-1}(t-t_j).\end{aligned} \tag{8.29}$$

将 $t=t_0$ 代入(8.29)式，得

$$\begin{aligned}P'_n(t_0)=&f[t_0,t_1]+f[t_0,t_1,t_2](t_0-t_1)+\\&f[t_0,t_1,t_2,t_3](t_0-t_1)(t_0-t_2)+\cdots+\\&f[t_0,t_1,t_2,\cdots,t_n](t_0-t_1)(t_0-t_2)(t_0-t_3)\cdots(t_0-t_{n-1}).\end{aligned} \tag{8.30}$$

(8.30)式右边的第 k 个部分和是根据前 k 个点的 k 阶牛顿插值多项式的导数. 可以得到下面的定理.

定理 8.5(牛顿插值多项式求导) 如果函数 $f(t)$ 在 $[a,b]$ 上可导，对于 $n+1$ 个节点 (t_j,y_j)，$j=0,1,\cdots,n$，其中 $t_j\in[a,b]$ 互不相同，满足 $f(t_j)=y_j$，$j=0,1,\cdots,n$. $t_0,t\in[a,b]$，$|t_0-t_1|\leqslant|t_0-t_2|\leqslant\cdots\leqslant|t_0-t_n|$，且 $\{(t_j,0)\}_{j=0}^{n}$ 形成在实数轴上

的 $n+1$ 个等距点，则有过 $n+1$ 个节点的 $f(t)$ 的 n 阶牛顿插值多项式 $P_n(t)$ 为 (8.28) 式，且 $P_n(t)$ 在 $t=t_0$ 处的导数 $P_n'(t_0)$，(8.30) 式的第 k 个部分和是精度为 $O(h^{k-1})$ 的 $f'(t_0)$ 的近似值.

我们利用数学归纳法，由 (8.30) 式可以推导出下面推论.

推论 8.1 如果函数 $f(t)$ 在 $[a,b]$ 上可导，对于 $n+1$ 个节点 (t_j,y_j)，$j=0,1,\cdots,n$，其中 $t_j\in[a,b]$ 互不相同，满足 $f(t_j)=y_j$，$j=0,1,\cdots,n$. $t_0,t\in[a,b]$，$|t_0-t_1|\leqslant|t_0-t_2|\leqslant\cdots\leqslant|t_0-t_n|$，且 $\{(t_j,0)\}_{j=0}^{n}$ 形成在实数轴上的 $n+1$ 个等距点，则 (8.30) 式的第 k 个部分和是精度为 $O(h^{k-1})$ 的 $f'(t_0)$ 的近似值，且当 $n=k$ 时.

(1) 如果 $t_j=x+jh$ $(j=0,1,2,\cdots,k-1)$，则 (8.30) 式是精度为 $O(h^{k-1})$ 的 $f'(x)$ 的 $k-1$ 阶前差公式.

(2) 如果 $t_j=x-jh$ $(j=0,1,2,\cdots,k-1)$，则 (8.30) 式是精度为 $O(h^{k-1})$ 的 $f'(x)$ 的 $k-1$ 阶后差公式.

(3) 如果 $t_0=x,t_1=x+h,t_2=x-h,t_3=x+2h,t_4=x-2h,\cdots$，则 (8.30) 式是精度为 $O(h^{k-1})$ 的 $f'(x)$ 的 $k-1$ 阶中心差商公式.

例如，(1) 当 $n=2$ 时，

① 如果 $t_0=x,t_1=x+h$，则 (8.30) 式为

$$f'(t_0)\approx P_n'(t_0)=f[t_0;t_1]=\frac{f(x+h)-f(x)}{h}$$

是精度为 $O(h)$ 的前差公式.

② 如果 $t_0=x-h,t_1=x$，则 (8.30) 式是精度为 $O(h)$ 的后差公式.

(2) 当 $n=3$ 时，

① 如果 $t_0=x,t_1=x+h,t_2=x+2h$，则 (8.30) 式为

$$f'(x)\approx P_n'(x)=\frac{4f(x+h)-f(x+2h)-3f(x)}{2h}$$

是精度为 $O(h^2)$ 的 $f'(x)$ 二阶前差公式.

② 如果 $t_0=x,t_1=x-h,t_2=x-2h$，则 (8.30) 式为

$$f'(x)\approx P_n'(x)=\frac{f(x-2h)-4f(x-h)+3f(x)}{2h}$$

是精度为 $O(h^2)$ 的 $f'(x)$ 二阶后差公式.

③ 如果 $t_0=x,t_1=x+h,t_2=x-h$，则 (8.30) 式为

$$f'(x)\approx P_n'(x)=\frac{f(x+h)-f(x-h)}{2h}$$

是精度为 $O(h^2)$ 的 $f'(x)$ 二阶中心差商公式.

(3) 当 $n=5$ 时，

① 如果 $t_j=x+jh\ (j=0,1,2,3,4)$，则(8.30)式是精度为 $O(h^4)$ 的 $f'(x)$ 的 4 阶前差公式.

② 如果 $t_j=x-jh\ (j=0,1,2,3,4)$，则(8.30)式是精度为 $O(h^4)$ 的 $f'(x)$ 的 4 阶后差公式.

③ 如果 $t_0=x,t_1=x+h,t_2=x-h,t_3=x+2h,t_4=x-2h$，则(8.30)式是精度为 $O(h^4)$ 的 $f'(x)$ 的 4 阶中心差商公式.

下面编写的 MATLAB 程序 diffnew.m 可以用来实现(8.30)式的计算.数据之间不需要等距，而且此程序只计算一点的导数 $f'(x_0)$. 程序 diffnew.m 是根据满足定理 8.5 条件的 $n+1$ 个节点 (x_j,y_j)，$j=0,1,\cdots,n$ 的差分求解，通过构造 n 阶牛顿插值多项式

$$\begin{aligned}P_n(x)=&f(x_0)+f[x_0,x_1](x-x_0)+\\&f[x_0,x_1,x_2](x-x_0)(x-x_1)+\cdots+\\&f[x_0,x_1,x_2,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1}),\end{aligned}$$

求解 $f'(x_0)$ 的近似值. 将 $f'(x_0)\approx P'(x_0)$ 作为最终结果. 如果在 x_k 处使用这个方法，则可以通过重新排列点的顺序为 $\{t_k,t_0,t_1,\cdots,t_{t-1},t_{t+1},\cdots,t_n\}$ 来计算 $f'(x_k)\approx P'(x_k)$.

利用牛顿插值多项式求导的 MATLAB 主程序

输入的量：X 和 Y 分别是由 $n+1$ 个节点 $(x_i,y_i)\ (i=1,2,\cdots,n+1)$ 的横坐标和纵坐标构成的向量；

输出的量：df 是 $f'(x_0)$ 的近似值，n 阶牛顿插值多项式 P 及其系数向量 A.

```
function [df,A,P] = diffnew(X,Y)
n = length(X);
A = Y;
for  j = 2:n
    for i = n:-1:j
        A(i) = (A(i) - A(i-1))/(X(i) - X(i-j+1));
    end
  end
    x0 = X(1);df = A(2);chsh = 1;m = length(A) - 1;
  for  k = 2:m
    chsh = chsh * (x0 - X(k));df = df + chsh * (A(k+1));
end
P = poly2sym(A);
```

例 8.2.8 试写出(8.30)式的精度为 $O(h^8)$ 的 $f'(x)$ 的 8 阶中心差商公式，并求 $f(x)=\sqrt{x}$ 在 $x=2$ 处导数的近似值，并与精确值比较，取 $h=0.1$.

解 根据推论8.1知,(8.30)式的精度为$O(h^8)$的$f'(x)$的8阶中心差商公式中的9个点分别为$x_0=2,x_1=2+h,x_2=2-h,x_3=2+2h,x_4=2-2h,x_5=2+3h,x_6=2-3h,x_7=2+4h,x_8=2-4h$.

输入程序

```
>> h=0.1;
X=[2,2+h,2-h,2+2*h,2-2*h,2+3*h;2-3*h,2+4*h,2-4*h];
Y=X.^(1/2);[df,A,P]=diffnew(X,Y),
df2=1/(2*2^(1/2)),wuq=abs(df - df2)
```

运行后屏幕显示$f'(2)$的精确值*df2*和近似值*df*及其绝对误差*wuq*,8阶中心差商公式*P*及其系数向量*A*如下

```
df =
    0.35355339040974
A =
   Columns 1 through 4
1.41421356237310  0.34924112245849  -0.04422874591121  0.01041424542736
   Columns 5 through 8
-0.00347548095288  0.00109476530972  -0.00046348001493  0.00015689880140
   Column 9
    -0.00007632355997
P =
2^(1/2)*x^8+3145684377932997/9007199254740992*x^7-
3187017017676797/ 72057594037927936.* x ^ 6 + 6003403753727975/
576460752303423488*x^5-4006956729431441/1152921504606846976*x^4+
631089234038427/ 576460752303423488 * x ^ 3 - 8549697218756559/
18446744073709551616*x^2+2894272034855853/18446744073709551616*
x-5631684710397231/73786976294838206464
df2 =                          wuq =
    0.35355339059327               1.835347429590684e-010
```

由上面输出的结果可见,用求导公式算出的精确值为3.535 533 905 932 737e-001与用(8.30)式的精度为$O(h^8)$的$f'(x)$的8阶中心差商公式计算的近似值3.535 533 904 097 390e-001的绝对误差为1.835 347 429 590 684e-010,所以取$h=0.1$时的计算结果精度就很高.

例8.2.9 根据下表给定的一组数据(X,Y)写出(8.30)式的具体形式及其精度和名称,并用它计算$f'(3.1352)$的近似值,取4位小数计算.

X	3.135 2	3.335 2	3.535 2	3.735 2	3.935 2	4.135 2	4.335 2	4.535 2
Y	0.126 6	-0.060 2	-0.603 2	-0.998 0	-0.119 4	0.995 3	-0.654 2	0.158 1

解 因为表中所给数据(X,Y)是等差数列,公差为 $h=0.2$,即

$$x_0=3.1352,x_1=x_0+h,x_2=x_0+2h,x_3=x_0+3h,$$
$$x_4=x_0+4h,x_5=x_0+5h,x_6=x_0+6h,x_7=x_0+7h.$$

根据推论 8.1 知,当 $n=8$ 时,(8.30)式是精度为 $O(h^7)$的$f'(x)$的 7 阶前差公式. 输入程序

```
>> X=[3.1352,3.3352,3.5352,3.7352,3.9352,4.1352,4.3352,4.5352 ];
 Y=[0.1266, -0.0602, -0.6032, -0.9980, -0.1194,0.9953, -0.6542
0.1581];
[df,A,P]=diffnew(X,Y)
```

运行后屏幕显示$f'(3.1352)$的近似值 df 和 7 阶牛顿插值多项式 P 及其系数向量 $\boldsymbol{A}$ 如下

```
df = -0.2428
A =
0.1266  -0.9340  -4.4525  10.5083  16.1667  -72.4818  64.7309  108.6155
P =
633/5000*x^7 -467/500*x^6 -1781/400*x^5 +1478916440133901/
140737488355328*x^4 +4550512123488957/281474976710656*x^3 -27833/
384*x^2 +4555032337958703/70368744177664*x +7643132912526197/
70368744177664
```

例 8.2.10 设函数$f(x)=\sin(5x^2-21)$. 利用(8.30)式的精度为 $O(h^4)$的$f'(x)$的 4 阶前差公式、4 阶后差公式和 4 阶中心差商公式计算$f'(3.1352)$的近似值,并与精确值比较,取 14 位小数计算,$h=0.001$.

解 (1) 根据推论 8.1 知,当 $n=5$ 时,① 如果 $x_0=3.1352,x_1=x_0+h,x_2=x_0+2h,x_3=x_0+3h,x_4=x_0+4h$,则(8.30)式是精度为 $O(h^4)$的$f'(x)$的 4 阶前差公式. ② 如果 $x_0=3.1352,x_1=x_0-h,x_2=x_0-2h,x_3=x_0-3h,x_4=x_0-4h$,则(8.30)式是精度为 $O(h^4)$的$f'(x)$的 4 阶后差公式. ③ 如果 $x_0=3.1352,x_1=x_0+h,x_2=x_0-h,x_3=x_0+2h,x_4=x_0-2h$,则(8.30)式是精度为 $O(h^4)$的$f'(x)$的 4 阶中心差商公式.

输入计算程序

```
>> x0=3.1352;h=0.001;Xq=[3.1352,x0+h,x0+2*h,x0+3*h,x0+4*h];
Xh=[3.1352, x0-h, x0-2*h, x0-3*h, x0-4*h];
Xz=[3.1352, x0+h, x0-h, x0+2*h,x0-2*h];
Yq=sin(5*Xq.^2-21);Yh=sin(5*Xh.^2-21);Yz=sin(5*Xz.^2-21);
[dfq,Aq,Pq]=diffnew(Xq,Yq),
[dfh,Ah,Ph]=diffnew(Xh,Yh),[dfz,Az,Pz]=diffnew(Xz,Yz)
syms x, f=sin(5.*x.^2-21);dy=diff(f,x)
```

运行后屏幕显示① 4 阶前差公式 P_q 和它的系数向量 A_q 及其 $f'(3.135\,2)$ 的近似值 df_q. ② 4 阶后差公式 P_h 和它的系数向量 A_h 及其 $f'(3.135\,2)$ 的近似值 df_h. ③ 4 阶中心差商公式 P_z 和它的系数向量 A_z 及其 $f'(3.135\,2)$ 的近似值 df_z，导数 $f'(x)$ 的符号形式分别如下

```
dfq =
    -31.09973888240658
Aq =
 1.0e+003 *
 Columns 1 through 4
0.00012659805395  -0.03116184195521  -0.05190522279697  5.11405450158927
 Columns 5
5.04316638931140
Pq =
  4561175588748639/36028797018963968*x^4 - 8771278738603423/
281474976710656*x^3 - 7305010688969305/140737488355328*x^2 +
2811481194788801/549755813888*x+5545020085856995/1099511627776
dfh =
    -31.09973905786443
Ah =
 1.0e+003 *
 Columns 1 through 4
0.00012659805395  -0.03102749818686  -0.08234817664867  5.02372948127046
 Column 5
8.97444729902917
Ph =
  4561175588748639/36028797018963968*x^4 - 8733464329536931/
281474976710656*x^3 - 5794737776087263/70368744177664*x^2 +
345228061216123/68719476736*x+5483510392960745/549755813888
dfz =
    -31.09974389667719
Az =
 1.0e+003 *
 Columns 1 through 4
0.00012659805395  -0.03116184195521  -0.06717188417315  5.08888712539359
 Column 5
7.53074172148027
Pz =
  4561175588748639/36028797018963968*x^4 - 8771278738603423/
```

```
281474976710656 * x ^ 3   4726801133312021/ 70368744177664 * x ^ 2 +
5595290566809831/1099511627776 * x +2070034522136353/274877906944
    dy -
        10 * cos(5 * x^2 -21) * x
```

(2) 为了将 $f'(3.1352)$ 的三种近似值与精确值比较,输入程序

```
>>  x -3.1352;dy  =10 * cos(5 * x^2 -21) * x,
wuq -dfq   ay ,wuh =dfh - dy,wuz =dfz - dy,
```

运行后屏幕显示 $f'(3.1352)$ 值及其三种近似值分别与精确值的差如下

```
dy  =                              wuq  =
    -31.09974488361178                  6.001205203887139e -006
wuh  =                             wuz  -
    5.825747354748501e -006             8.869345909407912e -007
```

由误差可见,用 4 阶中心差商公式计算 $f'(3.1352)$ 的近似值 df_z 与精确值的差 *wuz* 最小,用 4 阶前差公式计算 $f'(3.1352)$ 的近似值 df_h 与精确值的差 *wuh* 最大.

8.2.5 diff 函数在数值求导中的应用

在 MATLAB 系统中没有为用户提供数值求导的直接调用程序,但是我们可以利用系统中的差分函数 diff,或根据前面介绍过数值求导公式和截断误差公式,自己编写 MATLAB 程序来求给定函数 $y=f(x)$ 的数值导数,下面介绍 diff 函数在数值求导中的应用.

我们不但可以调用 diff 函数求一(多)元函数的符号(偏)导数,而且还可以计算差分,从而可以间接用 diff 求一元函数 $y=f(x)$ 的各阶数值导数或多元函数的数值偏导数. diff 调用格式和功能如表 8-7 和例题所示.

表 8-7 用 diff 计算数值求导和差分的调用格式和功能

数值求导的 MATLAB 命令	功 能
dX =diff(X)	(1) 如果 $\boldsymbol{X}$ 是向量,则 diff(X)输出 [X(2) - X(1),X(3) - X(2),…,X(n) X(n 1)]; (2) 如果 $\boldsymbol{X}$ 是矩阵,则 diff(X)输出矩阵 $\boldsymbol{X}$ 的行差分矩阵[X(2:n,:) - X(1:n - 1,:)].
yx =diff(y)./diff(x)	diff(y)./diff(x)实现用前差公式(8.2)求函数 $y=f(x)$ 对 x 的一阶数值导数 $y'=f'(x)\approx\frac{\Delta y}{\Delta x}$,其中函数 $y=f(x)$ 的增量 $\Delta y=$ diff(y),自变量 x 的增量 $\Delta x=$ diff(x).

续表

数值求导的 MATLAB 命令	功 能
dX =diff (X,N)	如果 X 是维数是 DIM 的向量或数组,且 N > = size(X,DIM),则 diff(X,N) 是 N 阶差分.
dX =diff (X,N,DIM)	如果 X 是向量或矩阵,则 diff(X,N,DIM) 是沿着 DIM 的 N 阶差分.如果 N > = size(X,DIM),则 diff 返回空的数组.

例如,(1) 取步长 $h=0.001$;自变量 x = 0:h:pi,则 diff(sin(x.^2))/h 的运行结果是 $\sin(x^2)$ 的导数 $2x\cos(x^2)$的近似值构成的数组;(2) diff((1:10).^2)的运行结果是 3:2:19.

例 8.2.11 设 $\boldsymbol{Y}=(0.1,2,3.2,4,5.6,8.9)$,依次求 $\boldsymbol{Y}$ 的一阶至四阶差分.

解 输入程序

```
>> Y =[0.1,2,3.2,4,5.6,8.9]; dY1 =diff(Y,1),dY2 =diff(Y,2),
dY3 =diff(Y,3),dY4 =diff(Y,4)
```

运行后屏幕显示 $\boldsymbol{Y}$ 的一阶至四阶差分如下

```
dY1 =
    1.9000    1.2000    0.8000    1.6000    3.3000
dY2 =
   -0.7000   -0.4000    0.8000    1.7000
dY3 =
    0.3000    1.2000    0.9000
dY4 =
    0.9000   -0.3000
```

例 8.2.12 设矩阵 $\boldsymbol{Y}=\begin{pmatrix}0 & 2 & 3 & 6\\ -1 & 21 & -15 & 7\end{pmatrix}$,依次求 dY1 = diff(Y,1), dY2 = diff(Y,2), dY3 = diff(Y,3),dY4 = diff(Y,4),diff(X,1,1), diff(X,1,2),diff(X,2,2), diff(X,3,2),并观察结果,总结 diff(Y,n) 和 diff(Y,n,m)的用法.

解 输入程序

```
>> Y =[0,2,3,6;-1,21,-15,7]; dY1 =diff(Y,1),
dY2 =diff(Y,2), dY3 =diff(Y,3) ,
dY4 =diff(Y,4), dY11 =diff(Y,1,1),
dY12 =diff(Y,1,2), dY22 =diff(Y,2,2), dY32 =diff(Y,3,2)
```

运行后屏幕显示 $\boldsymbol{Y}$ 的一阶至四阶差分如下

```
dY1 =                                   dY2 =
```

```
      -1    19   -18    1          20   -37   19
dY3 =                         dY4 =
     -57   56                    113
dY11 =                        dY32 =
      -1    19   -18    1      3  116
dY12 =                        dY22 =
     2      1     3               -1     2
     22   -36    22               -58   58
```

请读者自己总结 diff(Y,n)和 diff(Y,n,m)的用法.

例 8.2.13 设函数 $Y=f(x)$ 的自变量值 $\boldsymbol{X}=(2.800\ 2.900)$ 和函数值 $\boldsymbol{Y}=(1.0296\ 1.0979)$,用前差公式(8.2)求 $f'(2.800)$ 值.

解 输入程序

```
>> X=[2.800 2.900]; Y=[1.0296 1.0979]; YX=diff(Y)/diff(X)
```

运行后屏幕显示用前差公式(8.2)求 $f'(2.800)$ 值为

```
YX =
     0.6830
```



习 题 8.2

1. 设 $f(x)=\cos(15x^3-21)$.

(1) 分别利用前差公式和后差公式计算 $f'(0.79)$ 的近似值和误差,精度为小数点后 4 位,其中步长分别取 $h=0.1,0.01,0.001,0.0001$, $|f''(x)|\leqslant 80, x\in[0,1]$;

(2) 将(1)中计算的 $f'(0.79)$ 的近似值分别与精确值 $f'(0.79)$ 比较.

2. 设 $f(x)=\sin(3x^2+2x-6)$.

(1) 分别利用根据(8.7)式,(8.14)式,(8.23)式和(8.24)式计算 $f'(0.79)$ 的近似值,并估计误差,取小数点后 14 位计算,其中步长分别取 $h=0.1,0.01,0.001,0.0001$, $|f'''(x)|\leqslant M=872, x\in[0,1]$;

(2) 将(1)中计算的 $f'(0.79)$ 的近似值分别与精确值 $f'(0.79)$ 比较.

3. 当 $n=7$ 时,试由(8.30)式推导出前差公式、后差公式和中心差商公式.

4. 试用(8.30)式的精度为 $O(h^8)$ 的 $f'(x)$ 的 8 阶中心差商公式,求 $f(x)=\sqrt{x}$ 在 $x=3.2$ 处导数的近似值,取小数点后 14 位计算,h 取 0.01.

5. 给定一组数据 (X,Y) 如下表所示. 利用(8.30)式计算 $f'(2.1352)$ 的近似值,取 4 位有效数字,并判断所用公式的精度和名称.

X	2.135 2	3.335 2	3.535 2	3.735 2	3.935 2	4.135 2	4.335 2	4.535 2
Y	0.126 6	−0.060 2	−0.603 2	−0.998 0	−0.119 4	0.995 3	−0.654 2	0.158 1

6. 设函数$f(x)=\sin(4x^2-1)$. 利用(8.30)式的精度为$O(h^4)$的$f'(x)$的4阶前差公式、4阶后差公式和4阶中心差商公式计算$f'(5.45)$的近似值,并与精确值比较,取4位有效数字,取$h=0.01$.

7. 设矩阵$Y=\begin{pmatrix}-1 & 2 & 3 & 6\\ -12 & 21 & -15 & 7\end{pmatrix}$,依次求 dY1 = diff(Y,1),dY2 = diff(Y,2),dY3 = diff(Y,3),dY4 = diff(Y,4),diff(X,1,1),diff(X,1,2),diff(X,2,2),diff(X,3,2),并观察结果,总结 diff(Y,n)和 diff(Y,n,m)的用法.

8. 设函数$Y=f(x)$的自变量值$\boldsymbol{X}=(3.800\ 3.900)$和函数值$\boldsymbol{Y}=(11.029\,6\quad 21.097\,9)$,用前差公式(8.2)求$f'(3.800)$值.

9. 设函数$Y=f(x)$的值如下表,求$f'(2.8)$.

X	2.800 0	2.998 0	2.999 0	3.000 0	3.001 0	3.006 0	3.123 0	3.215 0
Y	1.029 6	1.097 9	1.098 3	1.098 6	1.098 9	1.100 6	1.138 8	1.167 8

10. 设$f(x)=\sin(8x^5-2x)$.

(1) 分别利用(8.7)式和(8.14)式计算$f'(1.79)$的近似值和误差,精度为小数点后4位,其中步长分别取$h=0.1,0.01,0.001,0.000\,1$;

(2) 将(1)中计算的$f'(1.79)$的近似值分别与精确值$f'(1.79)$比较.

11. 设一元函数$y=\arctan(5x^4-12\sin(6x^3-2))$.

(1) 求y的一阶导数和四阶导数,并化简;

(2) 求y的一阶微分.

12. 设已给出$y=f(x)$的数据如下表:

x	0.100 0	0.200 0	0.300 0	0.400 0	0.500 0	0.600 0	0.700 0
$f(x)$	1.221 4	1.491 8	1.822 1	2.225 5	2.718 3	3.320 1	4.055 2

$M=32.441\,7$,试用三点公式计算下列问题:

(1) 当$h=0.1$时,$y=f(x)$在$x=1.000\,0,1.100\,0,1.200\,0,1.300\,0,1.400\,0,1.500\,0,1.600\,0$处的一阶导数的近似值,并估计误差;

(2) 当$h=0.2$时,$y=f(x)$在$x=1.000\,0,1.200\,0,1.400\,0,1.600\,0$处的一阶导数的近似值,并估计误差;

(3) 当$h=0.3$时,$y=f(x)$在$x=1.000\,0,1.300\,0,1.600\,0$处的一阶导数的近似值,并估计误差;

(4) 表中的数据是函数$f(x)=e^{2x}$在相应点的数值,将(1)至(3)计算的一阶导数的近似值与$f(x)=e^{2x}$的一阶导数值比较,并求出它们的绝对误差.

8.3 高阶导数的数值计算及其 MATLAB 程序

高阶导数的数值计算是用离散方法近似地计算函数$y=f(x)$在某点x_0处的高阶导数值$f^{(n)}(x_0)$. 当然,通常只有函数以离散数值形式给出时才有必要用这

种方法.目前,常用的高阶导数的数值计算方法分为三类,第一类是利用插值或拟合求高阶数值导数方法;第二类是利用泰勒级数构造 n 阶导数的数值计算公式$P^{(n)}(x)$;第三类是利用高阶导数的定义构造 n 阶导数的数值计算公式$P^{(n)}(x)$.本节以范例的形式介绍前两类方法及其 MATLAB 程序.

8.3.1 插值或拟合求高阶数值导数方法及其 MATLAB 程序

插值或拟合求高阶数值导数方法就是利用第六章和第七章介绍的插值或拟合的方法,首先用给定的数值点构造一个插值函数或拟合函数 $P(x)$,然后或者将 $P(x)$的 n 阶导数 $P^{(n)}(x)$作为 n 阶导数的数值计算公式,或者对将 $P(x)$进行差分可得到 n 阶导数的数值计算公式,求函数$f(x)$的 n 阶数值导数$f^{(n)}(x_0)$的方法.下面以拉格朗日插值多项式为例介绍此类方法的应用及其用 MATLAB 的计算程序和算例.

(一) 插值或拟合求高阶数值导数方法

用下面的例题展示插值或拟合求高阶数值导数方法的思想和解题步骤.

例 8.3.1 给定 4 个点$(x_i,f(x_i))$ $(i=0,1,2,3)$.

(1) 利用拉格朗日插值多项式构造$f(x)$的一个 2 阶数值导数公式;

(2) 利用此公式求$f''(x_i)$ $(i=0,1,2,3)$的近似值;

(3) 如果 x_0,x_1,x_2,x_3成等差数列,公差为 h,求$f''(x_i)$ $(i=0,1,2,3)$的近似值.

解 (1) 首先构造$f(x)$的一个 2 阶数值导数公式.

根据(6.18)和(6.19)式可以得到过 4 个点 x_0,x_1,x_2,x_3的$f(x)$的 3 阶拉格朗日插值多项式为

$$L_3(x)=\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}f(x_0)+\frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}f(x_1)+\frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}f(x_2)+\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}f(x_3).$$

上式对 x 求二阶导数,得计算$f(x)$的一个 2 阶数值导数公式为

$$f''(x)\approx L''_3(x)=\frac{2[(x-x_1)+(x-x_2)+(x-x_3)]}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}f(x_0)+$$

$$\frac{2[(x-x_0)+(x-x_2)+(x-x_3)]}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}f(x_1)+$$

$$\frac{2[(x-x_0)+(x-x_1)+(x-x_3)]}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}f(x_2)+$$

$$\frac{2[(x-x_0)+(x-x_1)+(x-x_2)]}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}f(x_3). \tag{8.31}$$

(2) 利用(8.31)式求$f''(x_i)$ $(i=0,1,2,3)$的近似值.

将$x=x_i(i=0,1,2,3)$代入(8.31)式,得

$$f''(x_i)\approx\frac{2[(x_i-x_1)+(x_i-x_2)+(x_i-x_3)]}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}f(x_0)+$$

$$\frac{2[(x_i-x_0)+(x_i-x_2)+(x_i-x_3)]}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}f(x_1)+$$

$$\frac{2[(x_i-x_0)+(x_i-x_1)+(x_i-x_3)]}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}f(x_2)+$$

$$\frac{2[(x_i-x_0)+(x_i-x_1)+(x_i-x_2)]}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}f(x_3)\ (i=0,1,2,3). \tag{8.32}$$

(3) 如果x_0,x_1,x_2,x_3成等差数列,公差为h,由(8.32)式,得$f''(x_i)$ $(i=0,1,2,3)$的近似值为

$$f''(x_0)\approx\frac{2(-h-2h-3h)}{-h\cdot 2h\cdot 3h}f(x_0)+\frac{2(0-2h-3h)}{h\cdot h\cdot 2h}f(x_1)+$$

$$\frac{2(0-h-3h)}{-2h\cdot h\cdot h}f(x_2)+\frac{2(0-h-2h)}{3h\cdot 2h\cdot h}f(x_3),$$

即

$$f''(x_0)\approx\frac{2f(x_0)-5f(x_1)+4f(x_2)-f(x_3)}{h^2} \tag{8.33}$$

其余的$f''(x_i)$ $(i=1,2,3)$的近似值公式的推导留给读者.

(二) 拉格朗日插值多项式求高阶数值导数公式

如果函数$y=f(x)$必须取x_0的某一端点计算n阶数值导数$f^{(n)}(x)$,则不能用中心差商公式计算.在x_0的右(左)端点计算n阶数值导数$f^{(n)}(x_0)$用前差公式(后差公式),可以用拉格朗日多项式推导出这些公式.一些常用的高阶数值导数的前差公式和后差公式如表8-8所示,其中$f(x_0-kh)=f_{-k}$,$f(x_0+kh)=f_k$ $(k=1,2,\cdots,n)$.

表 8-8 常用的高阶数值导数的前差公式和后差公式

n 阶数值导数 $f^{(n)}(x_0)$ 前差公式	n 阶数值导数 $f^{(n)}(x_0)$ 后差公式
$f'(x_0)\approx\frac{-3f_0+4f_1-f_2}{2h}$	$f'(x_0)\approx\frac{3f_0-4f_{-1}+f_{-2}}{2h}$
$f''(x_0)\approx\frac{2f_0-5f_1+4f_2-f_3}{h^2}$	$f''(x_0)\approx\frac{2f_0-5f_{-1}+4f_{-2}-f_{-3}}{h^2}$
$f'''(x_0)\approx\frac{-5f_0+18f_1-24f_2+14f_3-3f_4}{2h^3}$	$f'''(x_0)\approx\frac{5f_0-18f_{-1}+24f_{-2}-14f_{-3}+3f_{-4}}{2h^3}$
$f^{(4)}(x_0)\approx\frac{3f_0-14f_1+26f_2-24f_3+11f_4-2f_5}{h^4}$	$f^{(4)}(x_0)\approx\frac{3f_0-14f_{-1}+26f_{-2}-24f_{-3}+11f_{-4}-2f_{-5}}{h^4}$

(三) 拉格朗日插值多项式高阶数值导数的 MATLAB 程序

按照(6.18)、(6.19)式编写了利用拉格朗日插值多项式构造 n 阶导数的数值计算公式的主程序,保存名为 ndaolag.m 的 M 文件.

利用拉格朗日插值多项式构造 n 阶导数的数值计算公式的 MATLAB 主程序

输入的量:$n+1$ 个节点(x_i,y_i) $(i=1,2,\cdots,n+1)$的横坐标向量 $\boldsymbol{X}$,纵坐标的向量 $\boldsymbol{Y}$,$f(x)$的最高阶导数 $f^{(n)}(x)$ 的阶数 n.

输出的量:n 次拉格朗日插值多项式 L 及其系数向量 $\boldsymbol{C}$,$f(x)$的 1 至 n 阶导数 $f^{(k)}(x)$ 的计算公式 dy_k $(k=1,2,3,\cdots,n)$和导数的阶数 k.

```
function [C,L,dyk,k] = ndaolag(X,Y,n)
m = length(X); n1 = m; L = ones(m,m);
for k = 1:m
    V = 1;
    for i = 1:m
     if k ~ = i
        V = conv(V,poly(X(i)))/(X(k) - X(i));
     end
    end
  L1(k,:) = V; l(k,:) = poly2sym (V);
end
C = Y * L1;L = Y *  l; syms x dyk
for k  = 1:n
      k;
dyk = diff(L,x,k)
end
```

例 8.3.2 给出节点数据 $f(-2.15)=17.03$,$f(-1.00)=7.24$,$f(0.01)=$

1.05，$f(1.02)=2.03$，$f(2.03)=17.06$，$f(3.25)=23.05$.

(1) 作五次拉格朗日插值多项式 L 和 $f(x)$ 的 1 至 5 阶数值导数公式；

(2) 利用此公式求 $y=f(x)$ 在 $x=-1.2345$ 处的 1 至 5 阶导数的近似值.

解 (1) 保存名为 ndaolag.m 的 M 文件.

(2) 在 MATLAB 工作窗口输入程序

```
>> X=[-2.15 -1.00 0.01 1.02 2.03 3.25];
Y=[17.03 7.24 1.05 2.03 17.06 23.05]; [C,L,dyk,k] = ndaolag(X,Y,5)
```

运行后屏幕显示五次拉格朗日插值多项式 L 及其系数向量 C，$f(x)$ 的 1 至 5 阶数值导数公式 dy_k（略）.

(3) 为了求 $y=f(x)$ 在 $x=-1.2345$ 处的 1 至 5 阶导数的近似值，输入程序

```
>> x = -1.2345;
dy1 = -26370266994304203933/5764607523034234880 +18224487282009991221/
2882303761517117440 * x^2 +59786195406624056511/230584300921369395200
* x^3 -12501150855594615669/11529215046068469760 * x^4 +2505830415074824099257/
368934881474191032320 * x
dy2 =18224487282009991221/1441151880758558720 * x +179358586219872169533/
230584300921369395200 * x^2  12501150855594615669/2882303761517117440
* x^3 +2505830415074824099257/368934881474191032320
dy3 =18224487282009991221/1441151880758558720 +179358586219872169533/
115292150460684697600 * x -37503452566783847007/2882303761517117440 * x^2
dy4 =179358586219872169533/115292150460684697600 -37503452566783847007/
1441151880758558720 * x
dy5 = -37503452566783847007/1441151880758558720
```

运行后屏幕显示 $y=f(x)$ 在 $x=-1.2345$ 处的 1 至 5 阶导数的近似值如下

```
dy1 =      dy2 =        dy3 =        dy4 =          dy5 =
   -6.3294     0.5262      -8.1043      33.6814        -26.0232
```

例 8.3.3 已知 $\sin 30°=0.5$，$\sin 45°=0.7071$，$\sin 60°=0.8660$，$\sin 75°=0.9659$，$\sin 90°=1$.

(1) 作四次拉格朗日插值多项式 L 和 $f(x)=\sin x$ 的 1 至 4 阶数值导数公式；

(2) 利用上面的公式求 $y=f(x)$ 在 $x=40°$处的 1 至 4 阶导数的精确值、近似值及其绝对误差，取小数点后 4 位和后 14 位计算.

解 (1) 取 $f(x)=\sin x$，则 $30°=\frac{\pi}{6}$，$45°=\frac{\pi}{4}$，$60°=\frac{\pi}{3}$，$75°=\frac{5\pi}{12}$，$90°=\frac{\pi}{2}$，求 $40°=\frac{2\pi}{9}$.

(2) 保存名为 ndaolag.m 的 M 文件.

（3）在 MATLAB 工作窗口输入程序

```
>> X =[pi/6,pi/4,pi/3,5*pi/12,pi/2];
Y =[0.5,0.7071,0.8660,0.9659,1];
[C,L,dyk,k] =ndaolag(X,Y,4), x =pi/6:pi/12:pi/2; y =sin(x);
[C1,L1,dyk1,k1] = ndaolag(x,y,4),
for i =1:4
  i,syms x,dyi =diff(sin(x),x,i)
end
```

运行后屏幕显示四次拉格朗日插值多项式 L 及其系数向量 C,$f(x)$的 1 至 4 阶数值导数 dy_k和符号导数 dy_i公式及其导数阶数 k 和 i.

（4）为了利用上面的公式求 $y=f(x)$在 $x=40°$处的 1 至 4 阶导数的近似值，并估计其误差，输入程序

```
>> x =2*pi/9;
dY1 =685769833743917463/703687441776640000 +1261982467915759/
10995116277760000 * x - 24314514941446017/ 35184372088832000 * x^2 +
6241572970666541/43980465111040000 * x^3
dY2 - 1261982467915759/10995116277760000 - 24314514941446017/
17592186044416000 * x +18724718911999623/43980465111040000 * x^2
dY3 = -24314514941446017/17592186044416000 +18724718911999623/
21990232555520000 * x
dY4 =18724718911999623/21990232555520000
dy1 - -6644262608911145859863501570909/79228162514264337593543950336
* x^3 +77172015474009052195694711806203/316912650057057350374175801344
* x^2 -34408318598267032320107245880583/158456325028528675187087900672 * x +
37371347371255248186645713466863/633825300114114700748351602688 +1/2 * 2^
(1/2) * ( - 2496629188266579/ 17592186044416 * x^3 + 16667207822766783/
35184372088832 * x^2 -8898045875479213/17592186044416 * x +6002949942623721/
35184372088832) +1/2 * 3^(1/2) * (7489887564799735/35184372088832 * x^3
11765087874894195/17592186044416 * x^2 +91239728215363/137438953472 * x -
7257297691828675/35184372088832)
dy2 = -19932787826733437579590504712727/79228162514264337593543950336
* x^2 +77172015474009052195694711806203/158456325028528675187087900672
* x -34408318598267032320107245880583/158456325028528675187087900672 +1/2
* 2^(1/2) * ( -7489887564799737/17592186044416 * x^2 +16667207822766783/
17592186044416 * x  8898045875479213/17592186044416) +1/2 * 3^(1/2) *
(22469662694399205/35184372088832 * x^2 -11765087874894195/
8796093022208 * x +91239728215363/137438953472)
dy3 = -19932787826733437579590504712727/39614081257132168796771975168
```

```
*x+77172015474009052195694711806203/158456325028528675187087900672+1/2
*2^(1/2)*(-7489887564799737/8796093022208*x+16667207822766783/
17592186044416)+1/2*3^(1/2)*(22469662694399205/17592186044416*x-
11765087874894195/8796093022208)
        dy4=-199327878267334375795905047127277/396140812571321687967719751 68
-7489887564799737/17592186044416*2^(1/2)+22469662694399205/
35184372088832*3^(1/2)
        dyi1=cos(x),dyi2=-sin(x),dyi3=-cos(x),dyi4=sin(x)
        WuY1=dyi1-dY1,WuY2=dyi2-dY2,WuY3=dyi3-dY3,WuY4=dyi4-dY4,
        wuy1=dyi1-dy1,wuy2=dyi2-dy2,wuy3=dyi3-dy3,wuy4=dyi4-dy4,
```

运行后将输出的 $y=f(x)$ 在 $x=40°$ 处的 1 至 4 阶导数（取小数点后 4 位和后 14 位）的近似值及其精确值，绝对误差的计算结果列入表 8－9 中.

表 8－9 例 8.3.3 的计算结果

导数阶数	取 4 位小数 $f'(40°)\approx dy_i$	取 14 位小数 $f'(40°)\approx dy_i$	精确值 $f'(40°)=dy_{ik}$
1	0.766 141 243 981 26	0.766 193 916 662 49	0.766 044 443 118 98
2	−0.642 619 475 133 15	−0.642 256 034 878 43	−0.642 787 609 686 54
3	−0.787 659 874 541 25	−0.790 355 501 585 45	−0.766 044 443 118 98
4	0.851 501 632 132 55	0.856 183 359 293 71	0.642 787 609 686 54
导数阶数	取 4 位小数的绝对误差	取 14 位小数的绝对误差	
1	−8.680 086 228 291 973e−005	−1.494 735 435 086 936e−004	
2	−1.681 345 533 890 433e−004	−5.315 748 081 105 021e−004	
3	0.021 615 431 422 27	0.024 311 058 466 47	
4	−0.208 714 022 446 01	−0.213 395 749 607 17	

由表 8－9 可见，$y=f(x)$ 在 $x=40°$ 处的 1 至 4 阶导数的（取小数点后 4 位比后 14 位）近似值的绝对误差小.

8.3.2 高阶泰勒数值导数方法及其 MATLAB 程序

所谓**高阶泰勒数值导数方法**就是利用泰勒级数构造 n 阶导数 $f^{(n)}(x)$ 的数值计算公式计算 $f^{(n)}(x_0)$ 的近似值的方法. 用下面的例题展示高阶泰勒数值导数方法的思想和解题步骤.

例 8.3.4 利用高阶泰勒数值导数方法证明 $f(x)$ 的精度为 $O(h^2)$ 的 2 阶数值导数公式为

$$f''(x_0)\approx\frac{f_1-2f_0+f_{-1}}{h^2}.$$

证明 $f(x_0\pm h)$ 在 x_0 处的泰勒展开式分别为

$$f(x_0+h)=f(x_0)+hf'(x_0)+\frac{h^2}{2}f''(x_0)$$

$$+\frac{h^3}{6}f'''(x_0)+\frac{h^4}{24}f^{(4)}(\xi_1), \tag{8.34}$$

$$f(x_0-h)=f(x_0)-hf'(x_0)+\frac{h^2}{2}f''(x_0)-\frac{h^3}{6}f'''(x_0)+\frac{h^4}{24}f^{(4)}(\xi_2). \tag{8.35}$$

将(8.34)式和(8.35)式相加，消去奇数阶导数项，得

$$f(x_0+h)+f(x_0-h)=2f(x_0)+\frac{2h^2}{2}f''(x_0)+\frac{h^4}{24}[f^{(4)}(\xi_1)+f^{(4)}(\xi_2)].$$

化简整理得

$$f''(x_0)=\frac{f(x_0+h)+f(x_0-h)-2f(x_0)}{h^2}-\frac{h^2}{24}[f^{(4)}(\xi_1)+f^{(4)}(\xi_2)]. \tag{8.36}$$

记截断误差为

$$R_3(f,h)=-\frac{h^2}{24}f^{(4)}(\xi), \tag{8.37}$$

其中 $f^{(4)}(\xi)=f^{(4)}(\xi_1)+f^{(4)}(\xi_2)$，则 4 阶泰勒数值导数公式为

$$f''(x_0)\approx\frac{f(x_0+h)+f(x_0-h)-2f(x_0)}{h^2}. \tag{8.38}$$

记作

$$f''(x_0)\approx\frac{f_1+f_{-1}-2f_0}{h^2}. \tag{8.39}$$

我们用同样的方法，可以推导出一些常用的精度为 $O(h^2)$ 和 $O(h^4)$ 的高阶数值导数的中心差商公式(如表 8－10 所示)，其中 $f(x_0-kh)=f_{-k}$，$f(x_0+kh)=f_k$ $(k=1,2,\cdots,n)$.

表 8－10 常用的精度为 $O(h^2)$ 和 $O(h^4)$ 的高阶数值导数的中心差商公式

精度为 $O(h^2)$ 的 n 阶数值导数 $f^{(n)}(x_0)$ 的中心差商公式	精度为 $O(h^4)$ 的 n 阶数值导数 $f^{(n)}(x_0)$ 的中心差商公式
$f'(x_0)\approx\frac{f_1-f_{-1}}{2h}$	$f'(x_0)\approx\frac{-f_2+8f_1-8f_{-1}+f_{-2}}{12h}$
$f''(x_0)\approx\frac{f_1+f_{-1}-2f_0}{h^2}$	$f''(x_0)\approx\frac{-f_2+16f_1-30f_0+16f_{-1}-f_{-2}}{12h^2}$
$f'''(x_0)\approx\frac{f_2-2f_1+2f_{-1}-f_{-2}}{2h^3}$	$f'''(x_0)\approx\frac{-f_3+8f_2-13f_1+13f_{-1}-8f_{-2}+f_{-3}}{8h^3}$
$f^{(4)}(x_0)\approx\frac{f_2-4f_1+6f_0-4f_{-1}+f_{-2}}{h^4}$	$f^{(4)}(x_0)\approx\frac{-f_3+12f_2-39f_1+56f_0-39f_{-1}+12f_{-2}-f_{-3}}{6h^4}$

例 8.3.5 设 $f(x)=\sin(5x^2-21)$.

(1) 分别利用(8.38)和(8.37)式计算 $f''(0.79)$ 的近似值和误差限、绝对误差和相对误差，精度为小数点后 4 位，其中步长分别取 $h=0.1,0.01,0.001,0.0001$，$|f^{(4)}(x)|\leqslant 9\,464$，$x\in[0,1]$；

(2) 将(1)中计算的$f''(0.79)$的近似值分别与精确值比较.

解 (1) 根据(8.38)和(8.37)式编写计算$y=f(x)$的二阶导数计算$f''(0.79)$的近似值和误差估计的MATLAB程序,并输入程序如下

```
>>  x =0.79;h =[0.1 0.01 0.001 0.0001];
M =9464;x1 =x +h;x2 =x -h; y =sin(5.*x.^2 -21);
y1 =sin(5.*x1.^2 -21); y2 =sin(5.*x2.^2 -21);
yz =(y1 +y2 -2 *y)./(h.^2), wu =abs( -h.^2.*M/24),
syms x,f =sin(5.*x.^2 -21); dy2 =diff(f,x,2)
```

运行后屏幕显示利用中心差商公式计算$f''(0.79)$的近似值 *yz* 和误差限 *wu*,精度为小数点后4位,其中步长分别取$h=0.1,0.01,0.001,0.0001$,$M=9464$,二阶导函数 *dy2* 如下

```
yz =
 45.02889719685589   -45.82347193518466    45.83048545869772
-45.83055543960768
wu =
  3.94333333333333   0.03943333333333   0.00039433333333
0.00000394333333
dy2 =
 -100 *sin(5 *x^2 -21) *x^2 +10 *cos(5 *x^2 -21)
```

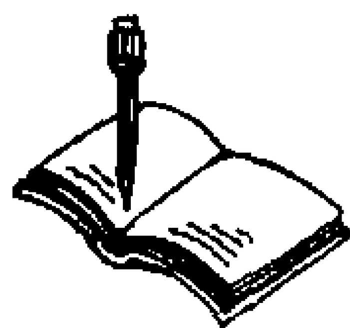
(2) 输入程序

```
>> x =0.79; dy2  = -100 *sin(5 *x^2 -21) *x^2 +10 *cos(5 *x^2 -21)
wuj =abs(yz -dy2), wux = wuj./abs(yz)
```

运行后屏幕显示利用中心差商公式计算$f''(0.79)$的近似值与精确值的绝对误差 *wuj*,相对误差 *wux* 和$f''(0.79)$的精确值 *dy2* 如下

```
dy2 =
      -45.83055620608437
wuj =
      0.80165900922847   0.00708427089971   0.00007074738664
 0.00000076647669
wux -
      0.01780321214006   0.00015459917375   0.00000154367526
 0.00000001672414
```

由(2)计算的结果可见,$h=0.0001$,利用中心差商公式计算$f''(0.79)$的近似值与精确值最为接近.



习题 8.3

1. 设$f(x)=\cos(15x^3-21)$.

(1) 分别利用(8.38)和(8.37)式计算$f''(0.79)$的近似值和误差限、绝对误差和相对误差,精度为小数点后 4 位,其中步长分别取 $h=0.1,0.01,0.001,0.0001$, $|f^{(4)}(x)|\leqslant 8\,454$, $x\in[0,1]$;

(2) 将(1)中计算的$f''(0.79)$的近似值分别与精确值比较.

2. 利用高阶泰勒数值导数方法证明表 8-10 中的$f(x)$的精度为$O(h^2)$和$O(h^4)$的 4 阶数值导数公式.

3. 利用高阶拉格朗日多项式数值导数方法证明表 8-10 中的$f(x)$的 4 阶数值导数公式.

4. 给出节点数据$f(-2.15)=17.03$,$f(-1.00)=7.24$,$f(0.01)=1.05$,$f(1.02)=2.03$,$f(2.03)=17.06$,$f(3.25)=23.05$.

(1) 作五次拉格朗日插值多项式 L 和$f(x)$的 1 至 5 阶数值导数公式;

(2) 利用此公式求 $y=f(x)$在 $x=-1.234\,5$ 处的 1 至 5 阶导数的近似值.

5. 已知 $\sin 30°=0.5$, $\sin 45°=0.707\,1$, $\sin 60°=0.866\,0$, $\sin 75°=0.965\,9$, $\sin 90°=1$.

(1) 作四次拉格朗日插值多项式 L 和$f(x)=\sin x$ 的 1 至 4 阶数值导数公式;

(2) 利用上面的公式求 $y=f(x)$在 $x=40°$处的 1 至 4 阶导数的取小数点后 4 位和后 14 位的近似值及其精确值,绝对误差.

6. 给定一组数据(X,Y)如下表所示.

X	2.135 2	3.335 2	3.535 2	3.735 2	3.935 2	4.135 2	4.335 2	4.535 2
Y	0.126 6	−0.060 2	−0.603 2	−0.998 0	−0.119 4	0.995 3	−0.654 2	0.158 1

(1) 作七次拉格朗日插值多项式 L 和$f(x)$的 1 至 6 阶数值导数公式;

(2) 利用此公式求 $y=f(x)$在 $x=3.234\,5$ 处的 1 至 6 阶导数的近似值,取 4 位有效数字.

7. 设函数 $f(x)=\sin(4x^2-1)$. 分别利用前差公式、后差公式和中心差商公式计算$f^{(8)}(5.45)$的近似值,并与精确值比较,取 4 位有效数字,取 $h=0.01$.

8. 设函数 $Y=f(x)$的值如下表,求$f^{(6)}(2.8)$.

X	2.800 0	2.998 0	2.999 0	3.000 0	3.001 0	3.006 0	3.123 0	3.215 0
Y	1.029 6	1.097 9	1.098 3	1.098 6	1.098 9	1.100 6	1.138 8	1.167 8

9. 设$f(x)=\sin(8x^5-2x)$.

(1) 利用中心差商公式计算$f^{(4)}(1.79)$的近似值和误差,精度为小数点后 4 位,其中步长分别取 $h=0.1,0.01,0.001,0.0001$.

(2) 将(1)中计算的$f'(1.79)$的近似值分别与精确值$f'(1.79)$比较.

8.4 数值梯度和数值偏导数的计算及其 MATLAB 程序

梯度和偏导数的数值计算可以用 MATLAB 系统中的 gradient 函数或

jacobian 函数完成. 另外,MATLAB 系统中还提供了用离散方法近似地计算拉普拉斯算子(Discrete Laplacian)的程序 DEL2 和雅可比行列式的程序 NUMJAC.本节介绍这些程序的功能和调用格式.

8.4.1 梯度和偏导数的数值计算及其 MATLAB 程序

在大学数学中,梯度是这样定义的:如果二元函数 $z=f(x,y)$ 在平面区域 D 内具有一阶连续偏导数,则对于每一点 $P(x,y)\in D$,都可以确定出一个向量,也就是梯度

$$\mathbf{grad}\, f(x,y)=\frac{\partial f}{\partial x}\boldsymbol{i}+\frac{\partial f}{\partial y}\boldsymbol{j},$$

其中$\frac{\partial f}{\partial x}$和$\frac{\partial f}{\partial y}$分别是 z 对 x 的偏导数和 z 对 y 的偏导数. MATLAB 系统中的 gradient 程序的主要功能是梯度、偏导数和导数的数值计算,详细的调用格式和功能如表 8-11 和例题所示.

表 8-11 gradient 程序的主要功能和调用格式

gradient 命令	功 能
[FX,FY] = gradient(F)	如果在每个方向上点与点之间的间距 H(即步长)是 1,$\boldsymbol{F}$ 是矩阵,则[FX,FY] = gradient(F)返回矩阵 $\boldsymbol{F}$ 的数值梯度. 其中 $\boldsymbol{FX}$ 相当于在 x(列)方向的数值偏导数$\frac{\partial F}{\partial x}$. $\boldsymbol{FY}$ 相当于在 y(行)方向的数值偏导数$\frac{\partial F}{\partial y}$. 当 $\boldsymbol{F}$ 是向量时,DF = gradient(F)返回 1-D 梯度.
[FX,FY] = gradient(F,H)	如果在每个方向上点与点之间的间距 H(即步长)相等,$\boldsymbol{F}$ 是矩阵或向量,则用此命令计算数值梯度.
[FX,FY] = gradient(F,HX,HY)	如果 $\boldsymbol{F}$ 是 2 维数组,且在 x 和 y 的向量或坐标点之间的间距分别是 HX 和 HY(即步长),$\boldsymbol{F}$ 是矩阵或向量,则用此命令计算数值梯度. 如果 $\boldsymbol{HX}$ 和 $\boldsymbol{HY}$ 是向量,则它们的长度必须与 $\boldsymbol{F}$ 对应的维数匹配.

续表

gradient 命令	功 能
[FX,FY,FZ] = gradient(F,H)	如果 F 是 3 维数组, H 是在每个方向上点与点之间的间距,则返回 F 的数值梯度. 其中 FX 相当于在 x 方向的数值偏导数 $\frac{\partial F}{\partial x}$. FY 相当于在 y 方向的数值偏导数 $\frac{\partial F}{\partial y}$. FZ 相当于在 z 方向的数值偏导数 $\frac{\partial F}{\partial z}$. 当 $H=1$ 时,[FX,FY,FZ] = gradient(F).
[FX, FY, FZ] = gradient (F, HX,HY,HZ)	如果间距由 HX,HY,HZ 给出,则用此命令计算数值梯度.
[FX, FY, FZ,...] = gradient (F,...)	同样地可以拓展到,即当 F 是 N 维数组,必须调用 N 个输出量和 2 或者 $N+1$ 个输入量时,则用此命令计算数值梯度.

例 8.4.1 设一元函数 $y = \arctan(3x^2+2\sin(5x^3-2))$, 求 y 在区间 $[0,3]$ 上,步长 $h=0.5$,每个分点上的一阶数值导数.

解 输入计算 y 对 x 的一阶数值导数的程序

```
>> x=0:0.5:3; y=atan(3*x.^2+2*sin(5*x.^3-2));
dy=gradient(y,0.5)
```

运行后屏幕显示 y 对 x 的一阶数值导数

```
dy =
    0.3744   2.3431   2.3307   0.2165   0.0712   0.0445   0.0300
```

例 8.4.2 设二元函数 $z=\arctan(3x^2+2\sin(5x^3-2y^5))+\dfrac{5x-6y-98}{7x^{13}+31y^{21}+9}$,取区域为 $0\leqslant x\leqslant 1, 0\leqslant y\leqslant 1.5$, x 和 y 的步长分别为 $Hx=0.2$ 和 $Hy=0.3$,试求各个分点上的 z 对 y 的数值偏导数 $\frac{\partial z}{\partial y}$.

解 输入计算求各个分点上的 z 对 y 的数值偏导数的程序

```
>> [x,y]=meshgrid(0:0.2:1, 0:0.3:1.5); % 坐标网格化
F=atan(3*x.^2+2*sin(5*x.^3-2*y.^5))+(5*x-6*y-98)./
(7*x.^13+31*y.^21+9);
[FX,FY]=gradient(F, 0.2, 0.3)
```

运行后屏幕显示各个分点上的 z 对 y 的数值偏导数如下

```
FY =
```

-0.6991	-0.6979	-0.6805	-0.6677	-0.6367	-0.3792
-1.1659	-1.1777	-0.9283	-0.6970	-0.6004	-0.4259
2.7988	2.4796	1.7458	3.5620	4.1588	1.7966
21.0134	20.5586	18.1694	18.0709	15.9847	10.7123
14.3933	14.2419	13.6587	11.0522	12.5391	8.9488
-6.0098	-6.0166	-6.3539	-6.1870	1.9905	0.5386

例 8.4.3 设二元函数 $z=3x^2e^{-x^2-y^2}$,取区域为 $-2.1\leqslant x\leqslant 2.1$, $-2.1\leqslant y\leqslant 2.1$,$x$ 和 y 的步长为 $Hx=Hy=0.2$,试求数值梯度向量,并画图.

解 输入计算数值梯度向量和画图的程序

```
>> [x,y] = meshgrid( -2.1:.2:2.1, -2.1:.2:2.1);
z = 3 * x.^2 .* exp( -x.^2 - y.^2); [px,py] = gradient(z,.2)
contour(z),% 曲面 z=3x^2e^(-x^2-y^2) 的等高线在 xoy 面上的投影线
hold on,quiver(px,py) % quiver 是二维向量场的表现函数
hold off
title('二维函数 z=3x^2exp( -x^2 -y^2)的数值梯度的图形')
```

运行后屏幕显示数值梯度 $\mathbf{grad}\, f(x,y)=\frac{\partial f}{\partial x}\boldsymbol{i}+\frac{\partial f}{\partial y}\boldsymbol{j}$ 向量(略)和图 8-2,其中 $\frac{\partial f}{\partial x}\approx px$,$\frac{\partial f}{\partial y}\approx py$ 的数据(略).

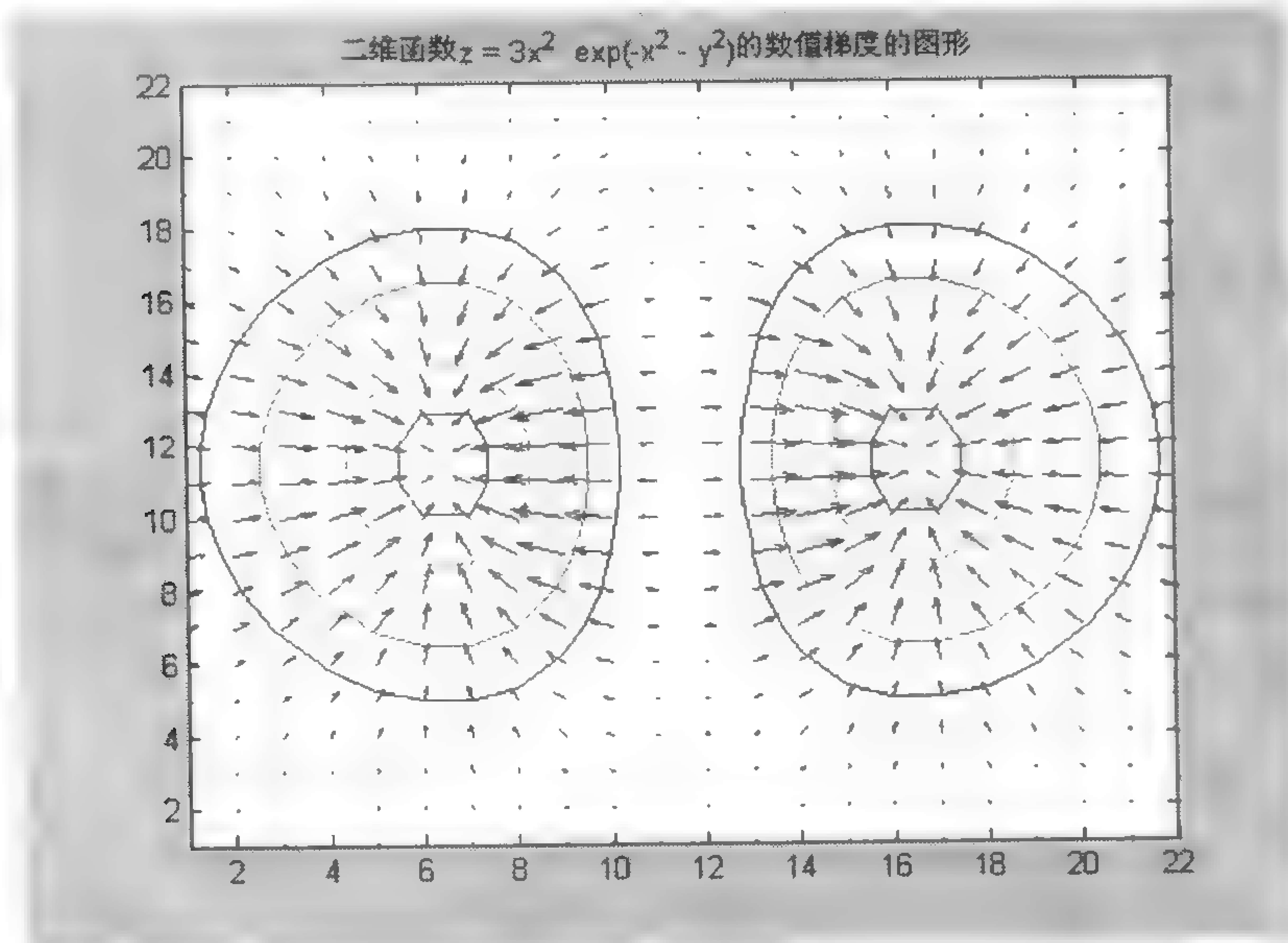


图 8-2 例 8.4.3 中 $z=3x^2e^{-x^2-y^2}$ 的数值梯度向量的图形

另外,在 MATLAB 系统中求曲面在每个节点的法向量的函数为surfnorm,详细的调用格式和功能如表 8-12 和例题所示.

表 8-12 求曲面在每个节点的法向量的函数 surfnorm 的调用格式和功能

符号求导的命令	功 能
[Nx,Ny,Nz] = surfnorm(X,Y,Z)	输入由(**X**,**Y**,**Z**)构成的 3 维数组,输出长度为 1 的被正规化的曲面的法向量的数值表达式[**Nx**,**Ny**,**Nz**].系统按(**X**,**Y**)计算步长
[Nx,Ny,Nz] = surfnorm(Z)	对于输入的曲面 Z 的函数值,输出曲面 Z 上的法向量的数值表达式[**Nx**,**Ny**,**Nz**].系统按 1 计算步长
surfnorm(X,Y,Z) 或 surfnorm(Z)	系统不输出值,而是自动先用 surf(Z)或 surf(X,Y,Z)画出曲面,然后在每个节点处以箭头的形式标出该点的法向量
surfnorm(...,'PropertyName ', Property-Value,...)	可以被用于设置指定曲面性质的向量,根据双三次拟合的数据输出曲面 Z 的法向量
surfnorm(X',Y',Z')	用于输出相反方向的曲面 Z 的法向量

例 8.4.4 设二元函数 $z=3(x-1)^2e^{-(x+1)^2-y^2}$,取区域为 $-2.1\leqslant x\leqslant 2.1$,$-2.1\leqslant y\leqslant 2.1$,$x$ 和 y 的步长为 $Hx=Hy=0.2$,试求该曲面在每个节点的法向量,并画图.

解 输入求该曲面在每个节点的法向量和画图的程序

```
>>  [x,y] = meshgrid(-2.1:.2:2.1, -2.1:.2:2.1);
z = 3*(x-1).^2 .* exp(-(x+1).^2 - y.^2);
[Nx,Ny,Nz] = surfnorm(x,y,z), surfnorm(x,y,z)
```

运行后屏幕显示该曲面在每个节点的法向量的数值表达式[Nx,Ny,Nz](略)和图 8-3.

8.4.2 计算雅可比矩阵及其行列式的 MATLAB 程序

在大学数学中,雅可比行列式是这样定义的.

定义 8.1 设二元函数 $u=f_1(x,y)$和 $v=f_2(x,y)$在平面区域 $\boldsymbol{D}$ 内具有一阶连续偏导数,由这些偏导数组成的行列式

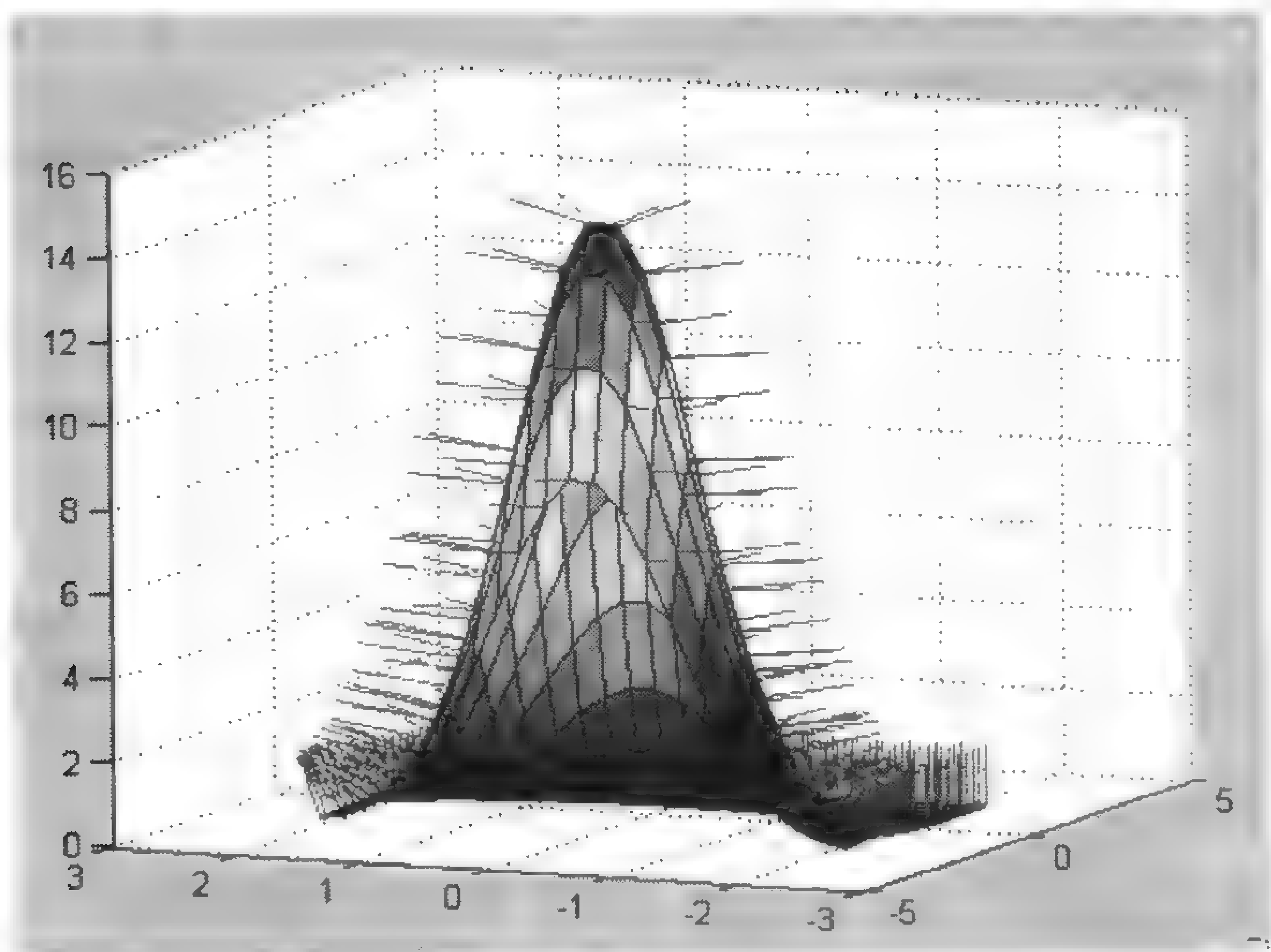


图 8－3 曲面 $z=3(x-1)^2e^{-(x+1)^2-y^2}$ 及其在每个节点的法向量的图形

$$\begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{vmatrix}$$

称作函数 u,v 关于自变量 x,y 的函数行列式或称**雅可比行列式**，记作

$$J=\frac{\partial(u,v)}{\partial(x,y)} \text{或} J=\frac{\partial(f_1,f_2)}{\partial(x,y)}.$$

可以将这个概念推广到含有 n 个自变量的 n 个函数的情形. 设 n 元 n 维向量函数 $f=(f_1,f_2,\cdots,f_m)$ 的每一个分量 $f_i=f_i(x_1,x_2,\cdots,x_n)$ $(i=1,2,\cdots,m)$ 在区域 $\boldsymbol{\Omega}$ 内具有一阶连续偏导数，由这些偏导数组成的行列式

$$J=\frac{\partial(f_1,f_2,\cdots,f_n)}{\partial(x_1,x_2,\cdots,x_n)}=\begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{vmatrix}$$

称作 n 元 n 维向量函数 $f=(f_1,f_2,\cdots,f_m)$ 的**雅可比行列式**.

雅可比行列式在重积分的换元法，隐函数存在定理等问题中都有应用. MATLAB 系统中提供了用离散方法近似地计算雅可比行列式的函数 numjac. 另外，还提供了符号计算雅可比矩阵的函数 jacobian. 下面分别介绍它们的功能和调用格式.

(一) 符号计算雅可比矩阵及其行列式

MATLAB 系统中提供了符号计算雅可比矩阵的函数 jacobian. 它的功能和调用格式如表 8-13 和例题所示.

表 8-13　符号计算雅可比矩阵的函数 jacobian

符号求雅可比矩阵的命令	功　能
Jfv = jacobian (f,v)	jacobian (f,v)是计算关于向量 v 的向量或标量 f 的雅可比矩阵 Jfv. 输出结果第(i,j)的表值是 $df(i)/dv(j)$. 注意：当 f 是标量时，f 的雅可比矩阵是 f 的梯度. 例如，对于一个函数 $u=f(x,y,z)$，则 jacobian(f) 求出 $f(x,y,z)$ 的梯度，它的第 1、2、3 个分量依次是 u 分别对 x,y,z 的偏导数. 另外，值得注意的是：如果 v 是标量，则 jacobian (f,v) = diff(f,v).

例 8.4.5　设 $\begin{cases} u=3(x-1)^2e^{-(x+1)^2-y^2}, \\ v=x^2-2x+y^2+2z+1, \\ w=3xyz, \end{cases}$ 求在 $x=2,y=1,z=5$ 处的雅可比矩阵和雅可比行列式 $J\left|_{\substack{x=2\\y=1\\z=5}}=\dfrac{\partial(u,v,w)}{\partial(x,y,z)}\right|_{\substack{x=2\\y=1\\z=5}}$.

解　(1) 首先求雅可比矩阵.

在 MATLAB 工作窗口输入程序

```
>> syms x y z
T = [x y z]; u = 3 * (x - 1)^2 * exp( - (x + 1)^2 - y^2);
v = x^2 - 2 * x + y^2 + 2 * z + 1; w = 3 * x * y * z;
f = [u;v;w]; Jfv = jacobian (f,T)
```

运行后输出 f 雅可比矩阵 Jfv 如下

```
Jfv =
[6 * (x - 1) * exp( - (x + 1)^2 - y^2) + 3 * (x - 1)^2 * ( - 2 * x - 2) *
exp( - (x + 1)^2 - y^2), - 6 * (x - 1)^2 * y * exp( - (x + 1)^2 - y^2), 0]
[2 * x - 2, 2 * y, 2]
[3 * y * z, 3 * x * z, 3 * x * y]
```

(2) 再求雅可比行列式 Jfd_{215}，输入程序

```
>> x = 2;y = 1;z = 5;
```

```
Jfv=[6*(x-1)*exp(-(x+1)^2-y^2)+3*(x-1)^2*(-2*x-
2)*exp(-(x+1)^2-y^2),-6*(x-1)^2*y*exp(-(x+1)^2-y^2),0;2*x
-2,2*y,2;3*y*z,3*x*z,3*x*y],
Jfd215=det(Jfv)
```

运行后输出 f 在 $x=2,y=1,z=5$ 处的雅可比矩阵 ***Jfv*** 及其行列式 Jfd_{215} 如下

```
Jfv =
  -0.00054479915715  -0.00027239957857                 0
   2.00000000000000   2.00000000000000  2.00000000000000
  15.00000000000000  30.00000000000000  6.00000000000000
Jfd215 =
  0.02124716712884
```

例 8.4.6 设 $u=3(x-1)^2e^{-(x+1)^2-y^2}+3xyz+x^2-2x+y^2+2z+1$.

求(1) u 的一阶偏导数,梯度和全微分;

(2) u 沿向量 $V=(2,1,5)$ 的方向导数;

(3) u 在 $x=2,y=1,z=5$ 处的一阶偏导数、梯度和全微分.

解 (1) 输入程序

```
>> syms x y z i j k dx dy dz
T=[x y z];
f=3*(x-1)^2*exp(-(x+1)^2-y^2)+3*x*y*z+x^2-2*x+y^
2+2*z+1;
Jfv=jacobian(f,T); Df= Jfv*[dx dy dz]',
ux= Jfv(1), uy= Jfv(2), uz= Jfv(3),
d=[i j k]; gradf=Jfv*d',V=[2,1,5]; Fxdf=Jfv*V'
```

运行后屏幕显示 u 沿向量 $V=(2,1,5)$ 的方向导数 *Fxdf*, u 的一阶偏导数 ux,uy,uz,梯度 *gradf* 和全微分 *Df*(略).

(2) 输入程序

```
>> syms i j k dx dy dz,x=2;y=1;z=5;
Df=(6*(x-1)*exp(-(x+1)^2-y^2)+3*(x-1)^2*(-2*x-2)
*exp(-(x+1)^2-y^2)+3*y*z+2*x-2)*conj(dx)+(-6*(x-1)^2*y*
exp(-(x+1)^2-y^2)+3*x*z+2*y)*conj(dy)+(3*x*y+2)*conj(dz)
ux=6*(x-1)*exp(-(x+1)^2-y^2)+3*(x-1)^2*(-2*x-2)*
exp(-(x+1)^2-y^2)+3*y*z+2*x-2
uy= -6*(x-1)^2*y*exp(-(x+1)^2-y^2)+3*x*z+2*y
uz=3*x*y+2
gradf= -i*(6*(x-1)*exp(-(x+1)^2-y^2)+3*(x-1)^2*(-
2*x-2)*exp(-(x+1)^2-y^2)+3*y*z+2*x-2)-i*(-6*(x-1)^2*y
*exp(-(x+1)^2-y^2)+3*x*z+2*y)+(3*x*y+2)*conj(k)
```

```
Fxdf =12 * (x -1) * exp( -(x +1)^2 -y^2) +6 * (x -1)^2 * ( -2 * x -
2) * exp( -(x +1)^2 -y^2) +6 * y * z +4 * x +6 -6 * (x -1)^2 * y * exp( -(x +1)
^2 -y^2) +3 * x * z +2 * y +15 * x * y
```

运行后屏幕显示 u 在 $x=2,y=1,z=5$ 处的沿向量 $V=(2,1,5)$ 的方向导数*Fxdf*，u 的一阶偏导数 *ux*，*uy*，*uz*，梯度 *gradf* 和全微分 *Df* 如下

```
Df =2392460628375541/140737488355328 * conj(dx) +9007122581075957/
281474976710656 * conj(dy) +8 * conj(dz)
ux =                      uy =                      uz =
  16.99945520084285         31.99972760042143          8
gradf = -13792043837827039/281474976710656 * i +8 * conj(k)
Fxdf =     1.059986380021071e +002
```

(二) 数值计算雅可比行列式

函数 numjac 的主要功能是数值计算函数 $F(T,Y)$ 的雅可比行列式 dF/dY，详细的调用格式和功能如表 8－14 和例题所示.

表 8－14　数值计算函数 $F(T,Y)$ 的雅可比行列式的函数 numjac 的调用格式和功能

数值计算雅可比行列式的命令	功　能
[DFDY,FAC] = numjac (F,T,Y,FTY, THRESH, FAC, VECTORIZED)	[DFDY,FAC] = numjac (F,T,Y,FTY,THRESH,FAC,VECTORIZED)的功能是数值计算函数 $F(T,Y)$ 的雅可比行列式 DFDY，返回 DFDY 的满矩阵. 其中 T 是自变量，Y 是包含因变量的列向量. 函数 F 必须返回一个列向量. 向量 *FTY* 是 F 在 (T,Y) 的估计. 列向量 *THRESH* 规定对于 Y 有意义的阈值，即满足 abs(Y(i)) < THRESH(i) 的 $Y(i)$ 的精确值. *THRESH* 的所有的元素必须是正的. 列向量 *FAC* 是工作存储器. 在第一次调用时，设置 *FAC* 到[　]. 在调用之间不改变返回值. VECTORIZED 识别 numjac 是否可以获得具有单个的函数计算的 F 的复合值. 尤其，VECTORIZED =1 表示 F(t,[y1 y2 ...]) 返回 [F(t,y1) F(t,y2) ...]，并且 VECTORIZED =2 表示 F([x1 x2 ...],[y1 y2 ...]) 返回[F(x1,y1) F(x2,y2) ...]. 当解 ODEs 时，如果 ODE 函数已经被编码，以便 F(t,[y1 y2 ...]) 返回[F(t,y1) F(t,y2) ...]，则用 ODESET 设置 ODE 解'Vectorized'的性质到'on'. 当解 BVPs 时，如果 ODE 函数已经被编码，以便 F([x1 x2 ...],[y1 y2 ...]) 返回[F(x1,y1) F(x2,y2) ...]，则用 BVPSET 设置 BVP 解'Vectorized'的性质到'on'. 向量化的函数 F 可以加速 DFDY 的计算.

续表

数值计算雅可比行列式的命令	功 能
[DFDY,FAC,G] = numjac (F,T,Y,FTY,THRESH,FAC,VECTORIZED,S,G)	[DFDY,FAC,G] = numjac (F,T,Y,FTY,THRESH,FAC,VECTORIZED,S,G) 的功能是数值地计算函数 $F(T,Y)$ 的雅可比行列式 *DFDY*,返回 *DFDY* 的稀松矩阵. *S* 是一个非空的 0 和 1 矩阵. 对于 $S(i,j)$ 来说,一个 0 意味着函数 $F(T,Y)$ 成分 i 不依赖于向量 Y 的成分 j(因此 $DFDY(i,j)=0$). 列向量 *G* 是工作储存器,在第一次调用时,设 *G* 到[ ],在调用之间不改变返回值.
[DFDY,FAC,G,NFEVALS,NFCALLS] = numjac (...)	[DFDY,FAC,G,NFEVALS,NFCALLS] = numjac(...) 返回当生成 DFDY (NFEVALS)时,被计算的值的次数和对于函数 F(NFCALLS)的调用的次数. 如果 *F* 没有被向量化,NFCALLS 等于 NFEVALS. 虽然,numjac 是特定地为 ODE's 积分时计算偏导数的近似值开发的程序,但是此程序也可以用于其他的应用. 尤其是由函数 $F(T,Y)$ 返回的长度不等于 *Y* 的长度时,*DFDY* 是长方形的.

例 8.4.7 设 $\begin{cases} u=-x^2+y, \\ v=x+y^2, \end{cases}$ 求 $J\bigg|_{\substack{x=1\\y=2}}=\dfrac{\partial(u,v)}{\partial(x,y)}\bigg|_{\substack{x=1\\y=2}}$.

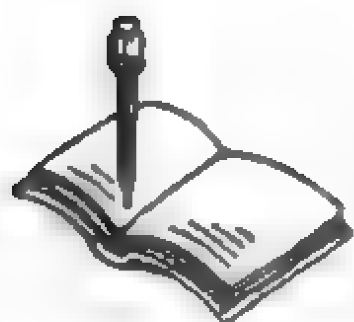
解 建立并保存名为 Y.m 和 F.m 的 M 文件

```
function Y = Y(T)
  x = T(1);y = T(2); Y = zeros(1,2); u = Y(1);
  Y(1) = -x^2 +y; v = Y(2); Y(2) = x +y^2;
function F = F(T,Y)
  T = zeros(2,1); x = T(1);y = T(2); F = zeros(2,2);
  F(1,1) = T(1);F(1,2) = T(2); F(2,1) = Y(1); F(2,2) = Y(2);
```

在 MATLAB 工作窗口输入程序

```
>> T = [1,2];y = Y(T); FTY = y;
THRESH = [10000,10000]; FAC = [ ]; VECTORIZED = 2;
numjac (@ F,T,@ Y, FTY, THRESH, FAC, VECTORIZED)
```

运行后输出结果 ans = -9.



习 题 8.4

1. 设一元函数 $y=e^{3x^2+2\sin(5x^3-2)}$,求 y 在区间[0,3]上,步长 $h=0.5$,分点上的一阶数值

导数.

2. 设二元函数 $z=\ln(9x^2+2\sin(5x^3-2y))+\dfrac{5x-6y-98}{7x^{13}+31y^{21}+9}$,取区域为 $0\leqslant x\leqslant 1,0\leqslant y\leqslant 1.5$,$x$ 和 y 的步长分别为 $Hx=0.2$ 和 $Hy=0.3$,试求各个分点上的 z 对 x 和 y 的数值偏导数 $\dfrac{\partial z}{\partial x}$ 和 $\dfrac{\partial z}{\partial y}$.

3. 设二元函数 $z=5x^2y^5\mathrm{e}^{-x^2-y^2}$,取区域为 $-2.1\leqslant x\leqslant 2.1$, $-2.1\leqslant y\leqslant 2.1$,$x$ 和 y 的步长为 $Hx=Hy=0.2$,试求数值梯度向量,并画图.

4. 设二元函数 $z=3x(y-1)^2\mathrm{e}^{-(x+1)^2-y^2}$,取区域为 $-2.1\leqslant x\leqslant 2.1$, $-2.1\leqslant y\leqslant 2.1$,$x$ 和 y 的步长为 $Hx=Hy=0.2$,试求该曲面在每个节点的法向量,并画图.

5. 设 $\begin{cases}u=2x(y-1)^2\mathrm{e}^{-(x+1)^2-y^2},\\ v=x^2-3x+y^2+4z+7,\\ w=5xyz,\end{cases}$ 求在 $x=2,y=3,z=5$ 处的雅可比矩阵和雅可比行列式.

6. 设 $u=3(x-1)^2\mathrm{e}^{-(x+1)^2-y^2}+3xyz+x^2-2x+y^2+2z+1$.

求(1) u 的一阶偏导数,梯度和全微分;

(2) u 沿向量 $V=(2,1,5)$ 的方向导数;

(3) u 在 $x=2,y=1,z=5$ 处的一阶偏导数、梯度和全微分.

7. 设 $\begin{cases}u=-x^2+y,\\ v=x+y^2,\end{cases}$ 求 $J\Big|_{\substack{x=1\\y=2}}=\dfrac{\partial(u,v)}{\partial(x,y)}\Big|_{\substack{x=1\\y=2}}$.

8. 设 $\begin{cases}u^2+v^2-x^2-y=0,\\ -u+v-xy+1=0,\end{cases}$ 求 $\dfrac{\partial v}{\partial x},\dfrac{\partial y}{\partial u}$.

9. 设 $\begin{cases}x=\mathrm{e}^u+u\sin v,\\ y=\mathrm{e}^u-u\cos v,\end{cases}$ 确定反函数组 $\begin{cases}u=u(x,y),\\ v=v(x,y),\end{cases}$ 求 $\dfrac{\partial u}{\partial x},\dfrac{\partial u}{\partial y},\dfrac{\partial v}{\partial x},\dfrac{\partial v}{\partial y}$.

第九章 数值积分

积分是非常重要的数学工具(微分方程、概率论等的基础),另一方面它们在实际问题中也有着许多直接的应用.函数的积分计算可分为数值积分和符号积分两类方法.数值积分是求积分的近似值的一类近似计算的方法.对于定积分 $\int_a^b f(x)\,dx$,如果对应的不定积分 $\int f(x)\,dx$ 不易求出,或者根本不能表示为初等函数时,那么就只能用数值积分的方法求其近似值了.例如,因为不定积分 $\int e^{-x^2}dx$, $\int \frac{\sin x}{x}dx$, $\int x^{-1}e^x dx$, $\int \frac{1}{\log_a x}dx$ 无法由初等函数表示,所以计算这种类型的定积分只能用数值方法.至于由离散数据或图形表示的函数的定积分,理所当然地属于数值积分的范畴.符号积分是指求积分的精确值.本章重点介绍数值积分及其用 MATLAB 计算的方法.为了将数值积分的值与精确值比较,在第一节中简单介绍用 MATLAB 做符号积分的方法.

9.1 积分的 MATLAB 符号计算

计算机软件 MATLAB 的系统提供了符号计算定积分的函数 int.下面主要介绍用 MATLAB 软件进行符号计算定积分、变上(下)限积分及其应用.

9.1.1 定积分的 MATLAB 符号计算

如果函数 $f(x)$ 在区间 $[a,b]$ 上连续,且 $F(x)$ 是 $f(x)$ 的一个原函数,则函数 $f(x)$ 在区间 $[a,b]$ 上的定积分为 $\int_a^b f(x)\,dx = F(b) - F(a)$.我们可以用 MATLAB 软件中的函数 int 对定积分进行符号计算,它的调用格式和功能如表 9-1 所示.

例 9.1.1 求 $\int_4^5 \frac{5}{(t-1)(t-2)(t-3)}dt$.

解 输入程序

```
>> syms t
f =5/((t -1)*(t -2)*(t -3)); F =int(f,t,4,5), y =double(F)
```

运行后屏幕显示计算定积分的值 F 及其近似值 y 如下

```
F =
```

```
   25/2*log(2)-15/2*log(3)
y =
    0.42474759198849
```

表 9-1 函数 int 求定积分的调用格式和功能

函数 int 求定积分的调用格式	功 能
F=int(S,a,b)	输入量:S 可以是被积函数 $f(x)$,也可以是由几个被积函数 $f_1(x),f_2(x),\cdots,f_n(x)$ 构成的矩阵,积分上、下限 a 和 b 可以是数,也可以是符号标量,积分变量是 x. 输出量:F 是 S 关于积分变量 x 的从 a 到 b 的定积分.
F=int(S,t,a,b)	输入量:S,a 和 b 同上,x 是积分变量. 输出量:F 是 S 关于积分变量 x 从 a 到 b 的定积分.

例 9.1.2 求 $\int_0^1 \frac{ax+b}{\sqrt[n]{cx+d}}\mathrm{d}x$.

解 输入程序

```
>> syms x a b c d n
f=(a*x+b)/((c*x+d)^(1/n)); F=int(f,0,1), y=simple(F)
```

运行后屏幕显示计算定积分化简后的值 y 如下

```
y =
    n*((-d^(1/n)*a*d^2+2*d^(1/n)*b*c^2+2*d^(1/n)*d*b*c+d^(1/n)*a*c^2-2*d*(c+d)^(1/n)*b*c+d^2*(c+d)^(1/n)*a)*n-d^(1/n)*c*d*a-d^(1/n)*d*b*c-d^(1/n)*b*c^2-d^(1/n)*a*c^2+d*(c+d)^(1/n)*b*c)/((c+d)^(1/n))/c^2/(-3*n+2*n^2+1)/(d^(1/n))
```

例 9.1.3 求下列定积分

(1) $\int_1^2 \frac{2x^2}{\sqrt{9-x^2}}\mathrm{d}x$; (2) $\int_1^2 \frac{\sqrt{x^2-1}}{2x}\mathrm{d}x$; (3) $\int_1^2 \frac{1}{\sqrt{(x^2+4)^3}}\mathrm{d}x$.

解 下面将三个定积分的被积函数 f_1,f_2,f_3 构成一个矩阵 $\boldsymbol{f}$,用一个积分命令求定积分,即输入程序

```
>> syms x
f1=(2*x^2)/(sqrt(9-x^2)); f2=sqrt(x^2-1)/(2*x);
f3=1/(sqrt((4+x^2)^3)); f=[f1; f2; f3];
F=int(f,1,2), y= double(F')
```

运行后屏幕显示三个定积分值的矩阵 $\boldsymbol{F}$ 和近似值矩阵 $\boldsymbol{y}$ 如下

```
F =
  [ -2*5^(1/2)+9*asin(2/3)+2*2^(1/2)-9*asin(1/3)]
```

```
[                              1/2*3^(1/2)-1/6*pi]
[                    1/8*2^(1/2)-1/20*5^(1/2)]
y =
      1.8653    0.3424    0.0650
```

即

$$\int_1^2 \frac{2x^2}{\sqrt{9-x^2}}\mathrm{d}x \approx 1.8653,\ \int_1^2 \frac{\sqrt{x^2-1}}{2x}\mathrm{d}x \approx 0.3424,\ \int_1^2 \frac{1}{\sqrt{(x^2+4)^3}}\mathrm{d}x \approx 0.0650.$$

例 9.1.4 已知$f(x)=\begin{cases}1+\sin x, & x\leqslant 1,\\ \frac{1}{2}x^2+5x-7, & x>1,\end{cases}$ 求$\int_0^2 f(x)\mathrm{d}x$.

解 输入程序

```
>> syms x
f1 =sin(x) +1; f2 =(x^2)/2 +5*x -7;
F =int(f1,x,0,1) + int(f2,x,1,2), y = double (F)
```

运行后屏幕显示计算的定积分值 F 和近似值 y 如下

```
F =
   -cos(1) +11/3
y =
   3.1264
```

例 9.1.5 由 $y=\sin x, y=\cos x, x=-\frac{1}{2}, x=\frac{3}{2}$所围成的平面区域 D,求平面区域 D 的面积 S.

解 (1) 画出由 $y=\sin x, y=\cos x, x=-\frac{1}{2}, x=\frac{3}{2}$所围成的封闭平面区域的图形,输入作函数图形的程序

```
>> x = -1:0.001:2; F1 = sin(x); F2 =cos(x);
plot(x ,F1,'b -',x ,F2,'g -'), axis([ -1,pi/4 +1, -1.3,1.3]),
xlabel('x'), ylabel('y'),
title('y =sinx , y =cosx 和 x = -0.5 及 x =1.5 所围成的平面区域的图
形')
```

运行后屏幕显示如图 9－1 所示.

(2) 求平面区域 D 的面积 S.

解方程组$\begin{cases}y=\sin x,\\ y=\cos x\end{cases}$得在$(-0.5,1.5)$内的两条曲线交点的横坐标 $x=\frac{\pi}{4}$,

所求面积为

$$S=\int_{-0.5}^{\frac{\pi}{4}}(\cos x-\sin x)\mathrm{d}x+\int_{\frac{\pi}{4}}^{1.5}(\sin x-\cos x)\mathrm{d}x.$$

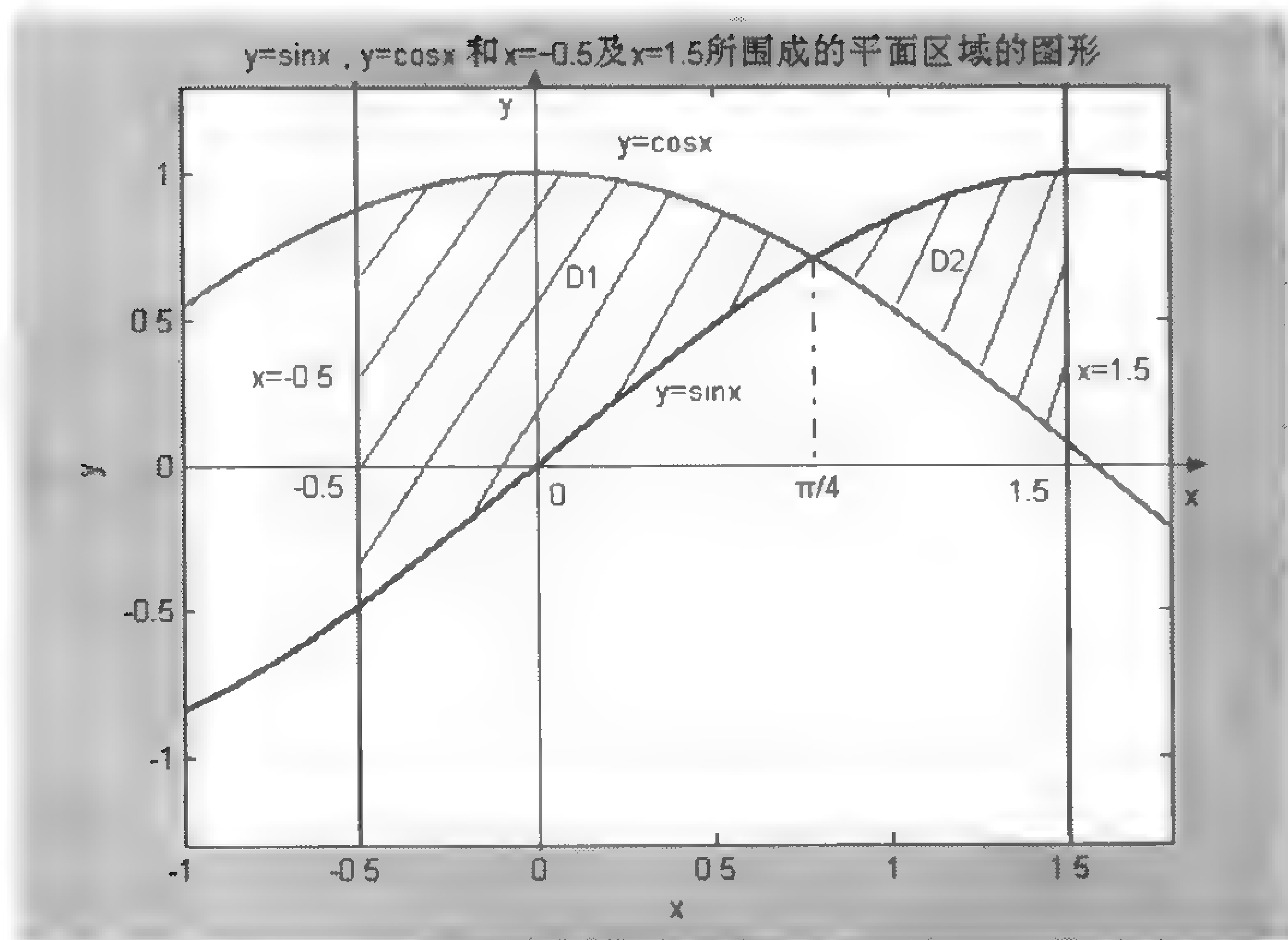


图 9-1 由 $y=\sin x, y=\cos x, x=-\frac{1}{2}, x=\frac{3}{2}$ 所围成的平面区域图形

输入计算面积 S 的程序

```
>> syms x
f1 = cos(x) - sin(x); f2 = - f1; S1 = int(f1,x, -0.5,pi/4);
S2 = int(f2,x, pi/4,1.5); S = S1 + S2,Sj = double (S)
```

运行后屏幕显示计算面积的值 S 及其近似值 Sj 如下

```
S =
    2*2^(1/2) + sin(1/2) - cos(1/2) - sin(3/2) - cos(3/2)
Sj =
    1.36203791318826
```

9.1.2 变限积分的 MATLAB 符号计算

例 9.1.6 求 $\Phi(x)=\int_{\sin x}^{2} t\sqrt{1+t^2}\mathrm{d}t$ 和导数 $\Phi'(x)$.

解 输入程序

```
>> syms x t
p = int(t*(sqrt(1+t^2)),sin(x),2), dp = diff(p)
```

运行后屏幕显示计算变下限积分 $\Phi(x)$ 和导数 $\Phi'(x)$ 的结果

```
p =
  5/3*5^(1/2) - 1/3*(1 + sin(x)^2)^(3/2)
```

```
dp =
  -(1+sin(x)^2)^(1/2)*sin(x)*cos(x)
```

例 9.1.7 已知 $F(x)=\int_x^{x^2} e^t\sin\left(2+\sqrt{t^3}\right)dt$，求 $F'(x)$.

解 因为

$$F'(x)=\left[\int_x^{x^2} e^t\sin\left(2+\sqrt{t^3}\right)dt\right]'$$
$$=\left[\int_x^{0} e^t\sin\left(2+\sqrt{t^3}\right)dt\right]'+\left[\int_0^{x^2} e^t\sin\left(2+\sqrt{t^3}\right)dt\right]',$$

所以输入程序

```
>> syms x t
F1=int(exp(t)*sin(2+sqrt(t^3)),x,0);
F2=int(exp(t)*sin(2+sqrt(t^3)),0,x^2);
Fi = F1 + F2;
dF=diff(Fi)
```

运行后屏幕显示计算变限积分 $F(x)$ 的导数 $F'(x)$ 如下

```
dF =
    -exp(x)*sin(2+(x^3)^(1/2))+2*x*exp(x^2)*sin(2+
(x^6)^(1/2))
```

例 9.1.8 求函数 $F(x)=\int_0^x 5t\cos(3t-2)dt$ 在区间[-1,3]上的最大值与最小值.

解 输入程序

```
>> syms x t
Fi=int((5*t)*cos(3*t-2),0,x)
```

运行后屏幕显示计算的变限积分 $F(x)$ 如下

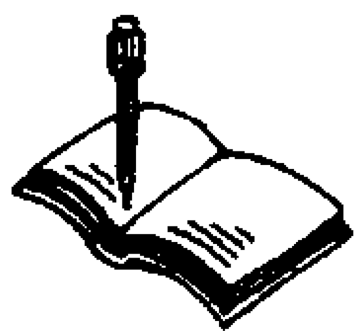
```
Fi =
  5/9*cos(3*x-2)+5/3*sin(3*x-2)*x-5/9*cos(2)
```

再输入程序

```
>> x=-1:0.001:3; Fi=5/9*cos(3*x-2)+5/3*sin(3*x-2).
*x-5/9*cos(2);
maxf=max(Fi), minf=min(Fi)
```

运行后屏幕显示变限积分 $F(x)$ 在区间[-1,3]上的最大值与最小值如下

```
maxf =
    3.93496026631086
minf =
   -3.49790870541011
```



习 题 9.1

用 MATLAB 软件完成下列问题，并用一个文件名存盘.

1. 计算下列积分.

(1) $\int_1^{e^2}\frac{1}{x\sqrt{1+\ln x}}dx$；　(2) $\int_1^{e}2x^3\ln^3 5x dx$；　(3) $\int_0^{\frac{1}{2}}\arcsin 3x dx$；

(4) $\int_0^{\frac{\pi}{4}}e^{-3x}\sin 2x dx$；　(5) $\int_0^{\frac{\pi}{4}}x^2\cos 2x dx$；　(6) $\int_0^{\frac{\pi}{8}}x\sec^2 2x dx$；

(7) $\int_0^{a}x^2\sqrt{a^2-x^2}dx$；　(8) $\int_0^{\sqrt{\ln 2}}x^3e^{x^2}dx$；　(9) $\int_1^{8}\frac{1}{\sqrt{2x-1}+\sqrt[3]{2x-1}}dx$.

2. 已知 $f(x)=\begin{cases}x+1, & x\leqslant 1,\\ \frac{1}{2}x^2, & x>1,\end{cases}$ 求 $\int_0^2 f(x)dx$.

3. 设 $f(x)=\begin{cases}2x+1, & |x|\leqslant 2,\\ 1+x^2, & 2<x\leqslant 4,\end{cases}$ 求 k 的值，使 $\int_k^3 f(x)dx=\frac{40}{3}$.

4. 求由函数 $y=f(x)=6x^5+\cos x$，$x=-2$，$x=3$ 和 $y=0$ 所围成的曲边梯形的面积，并画出它们的图形.

5. 求定积分 $\int_9^7\left(\cos^2\frac{ax}{2}+\frac{x^b}{35}\right)dx$，绘制其图形($a=2,b=3$)，并说明定积分的几何意义. 试探讨参数 a 和 b 对定积分的影响.

6. 设曲线 $y=f(x)$ 通过点(1,2)，且其切线的斜率为 $3x^2+2x-\frac{9}{1+x^2}$，求由此曲线 $y=f(x)$，$x=-2$，$x=3$ 和 $y=0$ 所围成的曲边梯形的面积，并画出它们的图形.

7. 求由曲线 $y=7-5x^3+3\log_2(2x+5)+7$，直线 $x=a$，$x=b$ ($a=-2,b=9$) 和 x 轴所围成的曲边梯形的面积，并画出它们的图形，试讨论 a,b 取不同的值时对曲边梯形的面积的影响.

8. 设曲线 $y=f(x)$ 通过点(2,3)，且其切线的斜率为 $3x^2+\sin x$，求由此曲线 $y=f(x)$，$x=a$，$x=b$ ($a=-2,b=8$) 和 $y=0$ 所围成的曲边梯形的面积，并画出它们的图形，试讨论 a,b 取不同的值时对曲边梯形的面积的影响.

9. 计算定积分 $\int_{-4}^5\left(\cos^2\frac{ax}{2}+5x^b+4\right)dx$，并用图形说明此定积分的几何意义($a=\pi,b=3$). 试探讨参数 a 和 b 对积分值的影响.

10. 求由抛物线 $y=3x^3+2x+3$，$x=a$，$x=b$ ($a=-4,b=1$) 和 x 轴所围成的平面图形的面积，并绘制其图形. 试探讨参数 a 和 b 对面积的影响.

9.2　数值积分的思想及其 MATLAB 程序

对于定积分 $\int_a^b f(x)dx$，如果被积函数 $f(x)$ 的原函数不易求出，或者根本不

能表示为初等函数时,那么就只能用定积分的数值方法求定积分的近似值了.例如,因为$\int \sin x^2 \mathrm{d}x$的被积函数的原函数无法表示为初等函数,计算这种类型的定积分只能用数值方法.有不少情况,被积函数$f(x)$没有具体的解析表达式,仅仅用表格或图形给出实验观测的一些点上的函数值,理所当然地属于数值积分的范畴.求定积分的数值计算的方法很多,这里主要介绍矩形公式、梯形公式、辛普森(Simpson)公式、一般的牛顿－科茨(Newton-Cotes)公式、递归公式和龙贝格(Romberg)公式、高斯－勒让德积分公式、拉道(Radau)积分公式、洛巴托(Lobatto)积分公式和它们的误差分析及其 MATLAB 程序.

9.2.1 数值积分的基本思想

用数值方法近似地求一个函数$f(x)$在闭区间$[a,b]$上定积分$I = \int_a^b f(x)\mathrm{d}x$的基本思路,可以归结到定积分的定义

$$I = \int_a^b f(x)\mathrm{d}x = \lim_{n\to\infty}\sum_{k=1}^{n} f(\xi_k)\frac{b-a}{n},$$

其中$\xi_k \in [x_{k-1},x_k] \subset [a,b]$ $(k=1,2,\cdots,n)$.当n的取值充分大时,I的数值积分就是

$$\int_a^b f(x)\mathrm{d}x \approx I_n = \sum_{k=1}^{n} f(\xi_k)\frac{b-a}{n}, \tag{9.1}$$

$$\int_a^b f(x)\mathrm{d}x = \sum_{k=1}^{n} f(\xi_k)\frac{b-a}{n} + R[f], \tag{9.2}$$

(9.1)和(9.2)式都称为**数值求积公式**,其中$R[f]$表示截断误差,称为**数值求积公式的余项**.如果数值求积公式(9.1)式对n次多项式精确地成立,则称此公式具有**n次代数精度**.显然,ξ_k取值不同,数值积分I_n的结果就不同,当然,定积分I的值是一样的.这种做法相当于用相对简单的阶梯函数$f(\xi_k)$,$k=1,2,\cdots,n$代替$f(x)$作积分.实际上各种不同的数值积分方法就主要在于,研究选择什么样的简单函数代替$f(x)$,使得既能保证一定的精度,计算量又小.

9.2.2 矩形公式

我们知道,如果在(a,b)上$f(x)\geqslant 0$,则定积分I表示曲线$f(x)$下的面积,不妨先从图形上看看如何近似计算这块面积(参见图 9－2).

将区间$[a,b]$进行n等分,积分步长为$h=(b-a)/n$.记

$$a = x_0 < x_1 < \cdots < x_k < \cdots < x_n = b.$$

在小区间上用矩形面积近似$f(x)$下面曲边梯形的面积,矩形公式中可以取左端点函数值为小矩形的高(图中的点线…),或取右端点函数值为小矩形的高(图

中的点划线 -.),于是在整个区间(a,b)内构成台阶形. 容易看出,两个台阶形面积分别为

$$L_n = h\sum_{k=0}^{n-1} f(x_k),\quad h=(b-a)/n, \tag{9.3}$$

$$R_n = h\sum_{k=1}^{n} f(x_k),\quad h=(b-a)/n. \tag{9.4}$$

在图 9-2 中,两个台阶形分别小于和大于所求面积. (9.3)和(9.4)式就是计算定积分的**矩形数值积分公式**,简称为**矩形公式**.

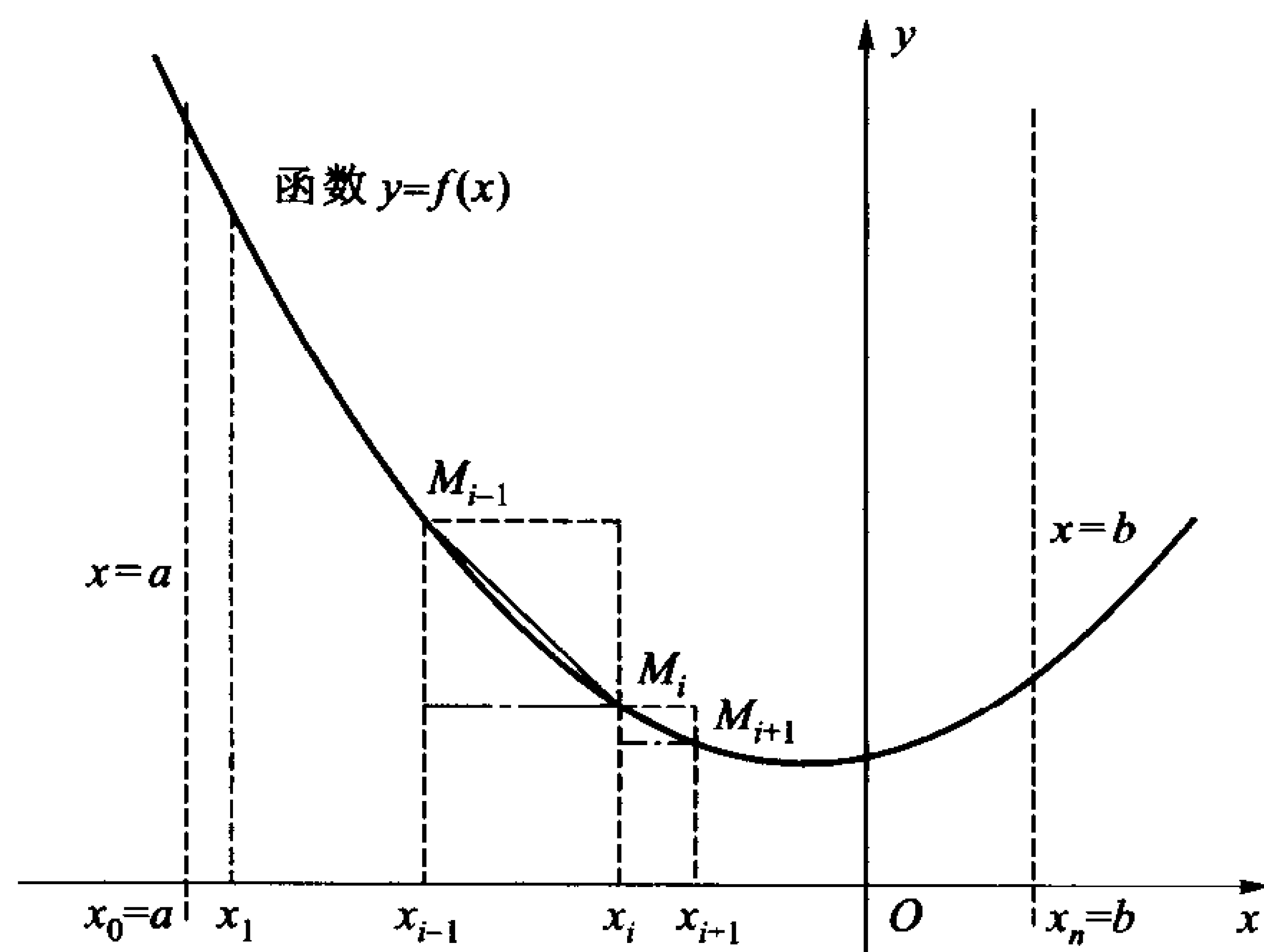


图 9-2 数值积分示意图

9.2.3 矩形公式的 MATLAB 程序

用矩形公式计算数值积分可用下面函数 sum 和 cumsum 实现,调用格式如下:

(一) 函数 sum 的调用格式

函数 sum 的两种调用格式如下:

调用格式一: sum(X)

如果输入向量 $\boldsymbol{X}$,则输出向量 $\boldsymbol{X}$ 的所有元的和,可用于按矩形公式(9.3)和(9.4)计算积分;如果输入矩阵 $\boldsymbol{X}$,则输出矩阵 $\boldsymbol{X}$ 的每列向量的元的和.

调用格式二: sum (X,DIM)

输入 N-D 数组 $\boldsymbol{X}$,输出为 $\boldsymbol{X}$ 的每个 DIM 向量的元的和.

例 9.2.1 分别计算矩阵 $\boldsymbol{X}=\begin{pmatrix}0&1&2\\3&4&5\end{pmatrix}$ 的行向量的元的和,列向量的元的和.

解 矩阵 X 的列向量的元的和的输入方法有两种 sum(X)和 sum(X,1)，输入程序

```
>> X = [0 1 2;3 4 5]; sh = sum(X), sh1 = sum(X,1), sL = sum(X,2)
```

运行后屏幕显示列向量的元的和 sh 或 sh_1，行向量的元的和 sL 结果如下

```
sh =                        sh1 =                      sL =
    3    5    7                 3    5    7                 3
                                                           12
```

例 9.2.2 用 MATLAB 和矩形公式(9.3),(9.4)计算 $\int_0^{\frac{\pi}{2}} e^{\sin x}dx$，并与精确值比较.

解 将 $\left[0,\frac{\pi}{2}\right]$ 分成 20 等份，步长为 $\frac{\pi}{40}$，输入程序

```
>> h = pi/40; x = 0:h:pi/2; y = exp(sin(x));
z1 = sum(y(1:20)) * h, z2 = sum(y(2:21)) * h,
```

运行后屏幕显示矩形公式计算结果分别如下

```
z1 =                   z2 =
    3.0364                 3.1713
```

即矩形公式(9.3)计算结果 $I\approx3.036\,4$，矩形公式(9.4)计算结果 $I\approx3.171\,3$.

求定积分的精确值，输入程序

```
>> syms x
F = int(exp(sin(x)),x,0, pi/2), Fs = double (F),
wz1 = abs( Fs - z1), wz2 = abs( Fs - z2)
```

运行后屏幕显示定积分的精确值 Fs 和与用矩形公式(9.3),(9.4)计算结果的绝对误差 wz_1、wz_2 分别如下

```
Warning: Explicit integral could not be found.
> In C:\MATLAB6p5p1\toolbox\symbolic\@sym\int.m at line 58
F =                                  Fs =
int(exp(sin(x)),x = 0 ..1/2*pi)          3.1044
wz1 =                                wz2 =
    0.0680                               0.0670
```

由此可见，该定积分用矩形公式(9.4)计算结果比矩形公式(9.3)的绝对误差小.

(二) 函数 cumsum 的调用格式

函数 cumsum 的调用格式如下：

调用格式一：cumsum(X)

输入向量 X，输出向量 X 的元依次累加和，可用于按矩形公式(9.3),(9.4)计算积分；

输入矩阵 X,输出矩阵为 X 的每列向量的元依次累加和.

调用格式二:cumsum (X,DIM)

输入 N-D 数组 X,输出为 X 的每个 DIM 向量的元依次累加和.

例 9.2.3 分别计算矩阵 $X=\begin{pmatrix}0&1&2\\3&4&5\\7&8&9\end{pmatrix}$的行向量的元的依次累加和,列向量的元的依次累加和.

解 矩阵 X 的列向量的元的和输入的方法有两种 cumsum (X)和 cumsum (X,1),输入程序

```
>> X = [0 1 2;3 4 5;7 8 9];
sh = cumsum (X), sh1 = cumsum (X,1), sL = cumsum (X,2)
```

运行后屏幕显示列向量的元的依次累加和 sh 或 sh_1,行向量的元的依次累加和 sL 结果如下

```
sh -                    sh1 =                    sL =
0     1     2           0     1     2            0     1     3
3     5     7           3     5     7            3     7     12
10    13    16          10    13    16           7     15    24
```

例 9.2.4 用 MATLAB 的函数 sum 和 cumsum 及矩形公式(9.3),(9.4)计算 $\int_0^{\frac{\pi}{2}} e^{-x}\sin x\mathrm{d}x$,并与精确值比较.

解 将 $\left[0,\frac{\pi}{2}\right]$ 分成 20 等份,步长为$\frac{\pi}{40}$,输入程序如下(注意 sum 和 cumsum 的用法)

```
>> h = pi/40; x = 0:h:pi/2; y = exp( -x) .* sin(x);
z1 - sum(y(1:20)) * h,z2 = sum(y(2:21)) * h,
z = cumsum(y); z11 - z(20) * h, z12 = (z(21) - z(1)) * h,
```

运行后屏幕显示计算结果分别如下

```
z1 =              z2 =              z11 =              z12 =
     0.3873            0.4036            0.3873             0.4036
```

求定积分的精确值,输入程序

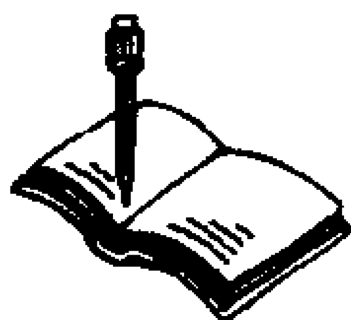
```
>> syms x
F = int(exp( -x) * sin(x),x,0, pi/2)
Fs = double (F) ,wz1 = abs( Fs - z1), wz2 = abs( Fs - z2)
```

运行后屏幕显示定积分的精确值 Fs 和用矩形公式(9.3),(9.4)计算结果的绝对误差 wz_1,wz_2分别如下

```
F -                                                 Fs =
     1/2 * ( -1 + exp(pi)^(1/2)) / exp(pi)^(1/2)         0.3961
```

```
wz1 =                                    wz2 =
    0.0088                                   0.0075
```

由此可见,用矩形公式(9.4)计算该定积分的结果比用矩形公式(9.3)计算结果的绝对误差小.



习 题 9.2

1. 分别计算矩阵 $X=\begin{pmatrix}0 & 1 & 2\\ 4 & 5 & 6\end{pmatrix}$ 的行向量的元的和及列向量的元的和.

2. 分别用 MATLAB 和矩形公式(9.3),(9.4)计算 $\int_0^{\frac{\pi}{2}} e^{\sin 2x}dx$,并与精确值比较.

3. 分别计算矩阵 $X=\begin{pmatrix}0 & 1 & 2\\ 23 & 24 & 25\\ 27 & 28 & 29\end{pmatrix}$ 的行向量的元的依次累加和及列向量的元的依次累加和.

4. 分别用 MATLAB 的函数 sum 和 cumsum 及矩形公式(9.3),(9.4)计算

$$\int_0^{\frac{\pi}{2}} e^{-2x}\sin 3x dx,$$

取 4 位有效数字,并与精确值比较.

9.3 插值型数值积分及其 MATLAB 程序

如果被积函数 $y=f(x)$ 和积分变量 x 是离散的数据 $X=(x_0,x_1,\cdots,x_n)$,$Y=(y_0,y_1,\cdots,y_n)$ 或数表,则常用的处理方法是利用各种插值函数(例如,样条插值函数、牛顿插值函数、拉格朗日插值多项式等)构造数值积分公式.

本节主要介绍利用拉格朗日多项式和样条插值函数等构造数值积分公式的方法、梯形公式、辛普森公式和一般的牛顿 - 科茨公式及其误差分析,三者之间的关系,并通过实例说明用 MATLAB 软件计算数值积分的方法.

9.3.1 梯形公式及其误差分析

若将矩形公式(9.3)和矩形公式(9.4)二者平均,则每个小区间上的小矩形变为小梯形(图 9 - 2 中细实线段 $M_{i-1}M_i$),整个区间上的结果可以用下面的定理表述.

定理 9.1(复合梯形公式) 设函数 $f(x)$ 在闭区间 $[a,b]$ 上的 $n+1$ 个等距节点 $x_k=a+kh$ $\left(k=0,1,2,\cdots,n,h=\dfrac{b-a}{n}\right)$ 处有定义,且其函数值为 $f(x_k)$

$(k=0,1,2,\cdots,n)$,则在闭区间$[a,b]$上存在

$$T_n = h\sum_{k=1}^{n-1} f(x_k) + \frac{h}{2}[f(x_0)+f(x_n)],\quad h=\frac{b-a}{n},\tag{9.5a}$$

或

$$T_n = \frac{h}{2}\sum_{k=1}^{n}[f(x_{k-1})+f(x_k)],\quad h=\frac{b-a}{n},\tag{9.5b}$$

使得

$$I=\int_a^b f(x)\,\mathrm{d}x = T_n + R(f,T_n),\tag{9.6}$$

其中$R(f,T_n)$是T_n的截断误差,即余项,(9.5a)和(9.5b)式就是常用的**复合梯形公式**,它相当于用分段线性插值函数作为$f(x)$的近似.

例 9.3.1 试用梯形公式($n=1$)计算定积分$\int_1^2 \mathrm{e}^{\frac{1}{x}}\mathrm{d}x$的值.

解 当$n=1$时,梯形公式为

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{2}[f(a)+f(b)].$$

取$a=1,b=2,f(x)=\mathrm{e}^{\frac{1}{x}}$,得

$$\int_1^2 \mathrm{e}^{\frac{1}{x}}\mathrm{d}x \approx \frac{1}{2}(\mathrm{e}-\sqrt{\mathrm{e}}) \approx 2.183\ 5.$$

例 9.3.2 分别取$h=\frac{\pi}{8\ 000},\frac{\pi}{800},\frac{\pi}{80}$,用梯形公式计算定积分$I=\int_0^{\frac{\pi}{2}} \mathrm{e}^{\sin x}\,\mathrm{d}x$,并与精确值比较.然后观察$h$对计算结果的有效数字和绝对误差的影响.

解 根据式(9.5a)编写并输入如下程序

```
>> h=pi/8000; a=0;b=pi/2; x=a:h:b;
n=length(x), y=exp(sin(x));
z1=(y(1)+y(n))*h/2; z2=sum(y(2:n-1))*h; z8000=z1+z2,
syms t
f=exp(sin(t)); intf=int(f,t,a,b), Fs=double(intf),
Juewucha8000=abs(z8000-Fs)
```

运行后屏幕显示取$h=\frac{\pi}{8\ 000}$时,积分区间$\left[0,\frac{\pi}{2}\right]$上等距节点的个数$n$,用梯形公式计算定积分$I$的值$z_{8\ 000}$和精确值*intf*的近似值$F$,及其绝对误差$Juewucha_{8\ 000}$如下

```
n =                              z8000 =
   4001                              3.10437900500451
Warning: Explicit integral could not be found.
> In C:\MATLAB6p5\toolbox\symbolic\@sym\int.m at line 58
```

```
intf =                                      Fs =
int(exp(sin(t)),t = 0 ..1/2*pi)   3.10437901785556
Juewucha8000 =
                         1.285104200832166e-008
```

然后再分别取 $h=\frac{\pi}{800},\frac{\pi}{80}$,运行后屏幕依次显示用梯形公式计算定积分 I 的值 z_{800},z_{80} 和与精确值 *intf* 的近似值 F_s 的绝对误差 $Juewucha_{800}$,$Juewucha_{80}$ 如下

```
n800 =        z800 =                  Juewucha800 -
    401         3.10437773275082        1.285104739068288e-006
n80 =          z80 -                  Juewucha80 =
   41           3.10425050738255       1.285104730022191e-004
```

由输出的结果可以看出,定积分 I 的被积函数的原函数不是初等函数,所以我们用 I 的近似值 $F_s=3.104\ 379\ 017\ 855\ 56$ 与用梯形公式计算 I 的值作比较. 取 $h=\frac{\pi}{8\ 000}$时,将积分区间$\left[0,\frac{\pi}{2}\right]$分成 4 000 等份,得 $z_{8\ 000}$ 与 F_s 的前八位数字相同,而第九位数字不同,且绝对误差 $Juewucha_{8000}=1.285\ 104\ 200\ 832\ 166\text{e}-008$,这说明 $z_{8\ 000}$ 具有八位有效数字. 故$I\approx3.104\ 379\ 0$. 取 $h=\frac{\pi}{800}$,得 z_{800} 与 F_s 的前六位数字相同,而第七位数字不同,且绝对误差 $Juewucha_{800}=1.285\ 104\ 739\ 068\ 288\text{e}-006$,这说明 z_{800} 具有六位有效数字. 故 $I\approx3.104\ 38$. 取 $h=\frac{\pi}{80}$,得 z_{80} 与 F_s 的前四位数字相同,而第五位数字不同,且绝对误差 $Juewucha_{80}=1.285\ 104\ 730\ 022\ 191e-004$,这说明 z_{80} 具有四位有效数字. 故 $I\approx3.104$.

由此可见,步长 h 越小,用梯形公式计算定积分 I 的结果具有的有效数字的位数就越多,绝对误差就越小,精度就越高. 下面的推论也给出了误差与步长 h 的关系.

推论 9.1(复合梯形公式的误差分析) 设函数 $f(x)$ 在闭区间 $[a,b]$ 上的 $n+1$ 个等距节点 $x_k=a+kh$ $\left(k=0,1,2,\cdots,n,h=\frac{b-a}{n}\right)$ 处的函数值为 $f(x_k)$ $(k=0,1,2,\cdots,n)$,且 $f(x)$ 在 $[a,b]$ 上具有连续的二阶导数,则**复合梯形公式**(9.5a)或(9.5b)是 2 **阶收敛**的,且存在 $\xi\in[a,b]$,使得式(9.5a)的截断误差 $R(f,T_n)$ 为

$$R(f,T_n)=\int_a^b f(x)\,\mathrm{d}x-T_n=-\frac{h^2(b-a)}{12}f''(\xi). \tag{9.7}$$

如果记 $M_2=\max\limits_{a\leqslant x\leqslant b}|f''(x)|$,$h=\frac{b-a}{n}$,则在 $[a,b]$ 上的绝对误差限为

$$|R(f,T_n)| \leqslant \frac{h^2}{12}M_2(b-a). \tag{9.8}$$

证明 梯形公式在小区间$[x_k,x_{k+1}]$上用拉格朗日线性插值函数 $L_1(x)$代替 $f(x)$，容易得到

$$f(x)=L_1(x)+\frac{f''(\xi_k)}{2}(x-x_k)(x-x_{k+1}),\ \xi_k\in[x_k,x_{k+1}].$$

因为$f''(\xi)$在$[a,b]$上连续，且$(x-x_k)(x-x_{k+1})\leqslant 0$，利用反常积分中值定理，在$[x_k,x_{k+1}]$上存在一点 $\xi_k\in[x_k,x_{k+1}]$，使误差为

$$\begin{aligned}\int_{x_k}^{x_{k+1}}[f(x)-L_1(x)]\mathrm{d}x &= \frac{f''(\xi_k)}{2}\int_{x_k}^{x_{k+1}}(x-x_k)(x-x_{k+1})\mathrm{d}x \\ &= -\frac{h^3}{12}f''(\xi_k).\end{aligned}$$

将所有小区间$[x_k,x_{k+1}]$上的误差相加，得

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x - T_n &= \sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}}[f(x)-L_1(x)]\mathrm{d}x \\ &= -\frac{h^3}{12}\sum_{k=0}^{n-1}f''(\xi_k).\end{aligned}$$

右项中的一个 h 用 $h=\dfrac{b-a}{n}$替换，得

$$\int_a^b f(x)\mathrm{d}x - T_n = -\frac{h^2(b-a)}{12}\left[\frac{1}{n}\sum_{k=0}^{n-1}f''(\xi_k)\right].$$

故存在 $\xi\in[a,b]$，使得

$$R(f,T_n)=\int_a^b f(x)\mathrm{d}x - T_n = -\frac{h^2(b-a)}{12}f''(\xi).$$

梯形求积公式(9.5a)的误差可用

$$|R(f,T_n)| = \left|\int_a^b f(x)\mathrm{d}x - T_n\right|. \tag{9.9}$$

进一步，记 $M_2=\max|f''(x)|, x\in(a,b)$，则

$$|R(f,T_n)| \leqslant \frac{h^2}{12}M_2(b-a). \tag{9.10}$$

(9.8)式表明，梯形公式(9.5a)的误差是 h^2阶的. 于是若$f\in C^2(a,b)$，由(9.8)，(9.9)式可得$\lim\limits_{n\to\infty}\dfrac{I-T_n}{h^2}=C$（非零常数），称梯形公式(9.5a)是 2 **阶收敛的**.

例 9.3.3 估计用梯形公式计算定积分 $I=\int_0^{\frac{\pi}{2}} \mathrm{e}^{\sin x}\mathrm{d}x$ 时的误差限，并与例

9.3.2 的误差进行比较，取 $h=\frac{\pi}{8\ 000}$.

解 根据估计误差公式(9.8)，即 $|R(f,T_n)|\leqslant\frac{h^2}{12}M_2(b-a)$，其中 $h=\frac{b-a}{n}$，$M_2=\max\limits_{a\leqslant x\leqslant b}|f''(x)|$，先输入求 $f''(x)$ 的程序

```
>>syms x
y=exp(sin(x)); yxx=diff(y,x,2)
```

运行后输出被积函数的二阶导函数. 然后输入误差估计程序如下

```
>> h=pi/8000; x=0:0.000001:pi/2;
yxx = -sin(x).*exp(sin(x))+cos(x).^2.*exp(sin(x));
RT=(h^2)*(pi/2)*max(yxx)/12
Wc= RT-(1.285104200832166e-008)
```

运行后屏幕显示误差限的估计值 *RT* 和 Wc = RT - Juewucha8000 的计算结果为

```
RT =                                  Wc =
     2.018637804707019e-008              7.335336038748535e-009
```

由此可见，误差限的估计值 *RT* 比例 9.3.2 的绝对误差 $Juewucha_{8000}$ 约大 7.336×10^{-9}，这符合推论 9.1.

9.3.2 梯形公式的 MATLAB 程序

用 MATLAB 软件和梯形公式计算数值积分和估计误差限除了用例 9.3.2 和例 9.3.3 自己编写的 MATLAB 计算程序外，MATLAB 系统提供了两种计算程序：一种是梯形数值积分(Trapezoidal doubleal integration)的程序 trapz.m；另一种是累加梯形数值积分(Cumulative trapezoidal doubleal integration)的程序 cumtrapz.m，它们的调用格式如下：

(一) 函数 trapz 的调用格式

调用格式一：Z =trapz(Y)

输入 ***Y***，输出为按梯形公式(9.5a)计算的 ***Y*** 的积分的近似值(单位步长).

(1) 如果输入 ***Y*** 是向量，则 ***Z*** 是 ***Y*** 的积分；

(2) 如果输入 ***Y*** 是矩阵，则行向量 ***Z*** 是 ***Y*** 的每列的积分.

调用格式二：Z =trapz(X,Y)

输入 ***X***，***Y*** 为同长度的数组，输出为按梯形公式(9.5a)计算 ***Y*** 对 ***X*** 的积分(步长不一定相等).

调用格式三：Z = trapz (X,Y,DIM)或 trapz (Y,DIM)

输出为按梯形公式(9.5a)计算 ***Y*** 的每个 DIM 的积分，其中 ***X*** 的长度必须和 size(Y,DIM)相同.

(二) 函数 cumtrapz 的调用格式

调用格式一：Z =cumtrapz (Y)

输入 Y,输出用梯形方法计算的 Y 的累加积分(Cumulative integral)的近似值(单位步长).

(1) 如果输入 Y 是向量,则 Z 是 Y 的累加积分；

(2) 如果输入 Y 是矩阵,则行向量 Z 是 Y 的每列的累加积分.

调用格式二：Z =cumtrapz (X,Y)

输入 X,Y 为同长度的数组,输出用梯形方法计算的 Y 对 X 的累加积分(步长不一定相等).

调用格式三：Z = cumtrapz (X,Y,DIM) 或 cumtrapz (Y,DIM)

输出用梯形方法计算的 Y 的每个 DIM 的累加积分,其中 X 的长度必须和 size(Y,DIM)相同.

例 9.3.4 用 MATLAB 的函数 trapz 和 cumtrapz 分别计算 $\int_0^{\frac{\pi}{2}} e^{-x}\sin x dx$, 精确到 10^{-4},并与矩形公式(9.3),(9.4)比较.

解 将$\left[0,\frac{\pi}{2}\right]$分成 20 等份,步长为$\frac{\pi}{40}$,输入程序如下(注意 trapz(y)是单位步长, trapz(y) * h = trapz(x,y))：

```
>> h=pi/40; x=0:h:pi/2; y=exp(-x).*sin(x);
z1 =sum(y(1:20))*h, z2 =sum(y(2:21))*h, z =(z1+z2)/2
z3 =trapz(y)*h, z3h =trapz(x,y), z3c =cumtrapz(y)*h,
```

运行后屏幕显示用矩形公式(9.3),(9.4)计算结果 z_1,z_2和二者的平均数 z,函数 trapz 和 cumtrapz 分别计算结果 z_3,z_{3c}分别如下

```
z1 =          z2 =          z =          z3 =          z3h =
    0.3873        0.4036        0.3954        0.3954        0.3954
z3c =
Columns 1 through 7
     0    0.0028    0.0109    0.0234    0.0395    0.0586    0.0798
Columns 8 through 14
0.1028    0.1270    0.1519    0.1771    0.2023    0.2273    0.2517
Columns 15 through 21
0.2755    0.2983    0.3201    0.3408    0.3602    0.3785    0.3954
```

显然,函数 trapz 和 cumtrapz 分别计算结果都与矩形公式(9.3),(9.4)计算结果的平均数 z 相等,即 $z_3=z_{3c}=z=0.3954$,梯形公式计算的结果与精确值 $Fs=0.3961$ 更接近. 但是,函数 cumtrapz 计算的结果展示了每次累加计算过程的结果.

例 9.3.5 用梯形公式(9.5a)的两种方法分别按列和行计算 $\int_0^n Y(x)\mathrm{d}x$,其中

(1) X 取 0 至 9 的整数,$Y=\begin{pmatrix}1 & 2 & 3\\ 11 & 22 & 31\end{pmatrix}$;

(2) $X=(0\quad 3\quad 11)$,$Y=(12\quad 22\quad 31)$.

解 (1) 将[0,9]分成9等份,步长为1,输入程序如下(注意 trapz(y)是单位步长)

```
>> Y=[1 2 3;11 22 31]; zL=trapz(Y,1), zh=trapz(Y,2),
zcL=cumtrapz(Y,1), zch=trapz(Y,2),
```

运行后屏幕显示用函数 trapz 和 cumtrapz 分别计算结果

```
zL = 6    12    17
zh =              zcL =                       zch =
  4                    0     0     0               4
  43                   6    12    17              43
```

(2) 步长不相等,输入程序

```
>> X=[0 3 11];Y=[12 22 31]; z=trapz(X,Y),zc=cumtrapz(X,Y)
```

运行后屏幕显示用函数 trapz 和 cumtrapz 分别计算结果如下:

```
z =              zc =
  263                 0    51   263
```

(三) 自编梯形数值积分的 MATLAB 主程序

另外,也可以根据梯形公式和连续增加的子区间数来逼近

$$\int_a^b f(x)\mathrm{d}x\approx\frac{h}{2}\sum_{k=1}^{2^m}[f(x_{k-1})+f(x_k)].$$

编写计算数值积分的程序如下:

梯形数值积分的 MATLAB 主程序
输入量:*fun* 是被积函数 $f(x)$,a,b 分别是积分下、上限,m 是递归的次数.
输出量:T 是利用梯形公式数值计算 $\int_a^b f(x)\mathrm{d}x$ 的递归值.

```
function T=rctrap(fun,a,b,m)
n=1;h=b-a; T=zeros(1,m+1); x=a;
T(1)=h*(feval(fun,a)+feval(fun,b))/2;
for i=1:m
  h=h/2; n=2*n; s=0;
  for k=1:n/2
    x=a+h*(2*k-1); s=s+feval(fun,x);
  end
```

```
  T(i+1)=T(i)/2+h*s;
end
T=T(1:m);
```

例 9.3.6 用 rctrap 计算 $I=\frac{1}{\sqrt{2\pi}}\int_0^{\frac{\pi}{2}} e^{-\frac{x^2}{2}}dx$,递归 14 次,并将计算结果与精确值比较.

解 (1) 建立 fun.m 文件命名的 M 文件函数

```
function y = fun(x)
        y = exp( ( -x.^2)./2)./(sqrt(2*pi));
```

(2) 输入程序

```
>>T=rctrap(@fun,0,pi/2,14), syms t
 fi=int(exp((-t^2)/2)/(sqrt(2*pi)),t,0, pi/2);
 Fs = double(fi), wT = double(abs(fi-T))
```

运行后屏幕显示 I 精确值 Fs,用 rctrap 计算 I 的递归值 T 和 T 与精确值 Fs 的绝对误差 w_T 如下

```
T =
  Columns 1 through 4
   0.40457385587044   0.43245899179539   0.43953669400491
   0.44129851884975
  Columns 5 through 8
   0.44173842995173   0.44184837274713   0.44187585624611
   0.44188272698315
  Columns 9 through 12
   0.44188444465881   0.44188487407718   0.44188498143174
   0.44188500827038
  Columns 13 through 14
   0.44188501498004   0.44188501665745
Fs =
   0.44188501721659
wT =
  Columns 1 through 4
   0.03731116134615   0.00942602542121   0.00234832321168
   0.00058649836684
  Columns 5 through 8
   0.00014658726486   0.00003664446946   0.00000916097048
   0.00000229023344
  Columns 9 through 12
   0.00000057255779   0.00000014313941   0.00000003578485
```

```
0.00000000894621
Columns 13 through 14
0.00000000223655   0.00000000055914
```

由此可见,递归14次,用 rctrap 计算数值积分的结果与精确值的绝对误差为 5.5914×10^{-10},$I\approx0.441885017$.

9.3.3 辛普森(Simpson)公式及其误差分析

为提高精度可以用分段二次插值函数代替 $f(x)$. 由于每段要用到相邻两个小区间端点的三个函数值,所以小区间的数目必须是偶数,记 $n=2m,k=0,1,\cdots,m$. 在第 k 段的两个小区间上用三个节点 $(x_{2k},f(x_{2k}))$,$(x_{2k+1},f(x_{2k+1}))$,$(x_{2k+2},f(x_{2k+2}))$ 作二次插值函数 $S_k(x)$, 然后积分可得

$$\int_{x_{2k}}^{x_{2k+2}} S_k(k)\mathrm{d}x=\frac{h}{3}[f(x_{2k})+4f(x_{2k+1})+f(x_{2k+2})],$$

求 m 段之和就得到整个区间上的近似积分.

定理 9.2(复合辛普森公式) 设函数 $f(x)$ 在闭区间 $[a,b]$ 上的 $2m+1$ 个等距节点 $x_k=a+kh\ \left(k=0,1,2,\cdots,2m,h=\dfrac{b-a}{2m}\right)$ 处有定义,且其函数值为 $f(x_k)$ $(k=0,1,2,\cdots,2m)$,则在 $[a,b]$ 上有

$$S_n=\frac{h}{3}\left[f(a)+f(b)+2\sum_{k=1}^{m-1}f(x_{2k})+4\sum_{k=1}^{m}f(x_{2k-1})\right],h=\frac{b-a}{2m} \quad (9.11)$$

使得

$$I=\int_a^b f(x)\mathrm{d}x=S_n+R(f,S_n), \quad (9.12)$$

其中 $R(f,S_n)$ 是 S_n 的截断误差,即余项. (9.11)式称为**复合辛普森数值积分公式**,简称**辛普森公式**.

例 9.3.7 用辛普森公式(9.11)计算 $I=\dfrac{1}{\sqrt{2\pi}}\int_0^1 \mathrm{e}^{-\frac{x^2}{2}}\mathrm{d}x$,取 $n=20\,001$ 个等距节点,并将计算结果与精确值比较,然后再取 $n=13$ 计算,观察 n 对误差的影响.

解 由 $n=2m+1=20\,001$,得 $m=10\,000$. 根据辛普森公式(9.11)编写并输入下面的程序

```
>> a=0;b=1;m=10000; h=(b-a)/(2*m); x=a:h:b;
y=exp((-x.^2)./2)./(sqrt(2*pi));
z1=y(1)+y(2*m+1); z2=2*sum(y(2:2:2*m)); z3=4*sum(y(3:2:2*m));
z=(z1+z2+z3)*h/3, syms t,f=exp((-t^2)/2)/(sqrt(2*pi));
intf=int(f,t,a,b), Fs=double(intf); Juewucha=abs(z-Fs)
```

运行后屏幕显示用辛普森公式(9.11)计算定积分 I 的近似值 z 和精确值 $intf$ 及

其绝对误差 *Juewucha*（取 $n=20\ 001$ 个等距节点）如下

```
z =
   0.34133406408431
intf =
1125899906842624/5644425081792261 *erf(1/2 *2^(1/2))*2^(1/2)*pi^(1/2)
Juewucha =
   1.068198423692657e-005
```

然后再取 $n=13$ 计算的近似值 z 和与精确值 *intf* 及其绝对误差 *Juewucha* 的结果为：$z=0.323\ 261\ 353\ 494\ 30$，$Fs=0.341\ 344\ 746\ 068\ 54$，$Juewucha=0.018\ 083\ 392\ 574\ 25$. 由此可见，节点的个数 n 越大，步长 h 越小，其绝对误差也越小. 这个结论我们可以用证明推论 9.1 的方法证明，其结果见下面的推论.

推论 9.2（复合辛普森公式的误差分析） 设函数 $f(x)$ 在闭区间 $[a,b]$ 上的 $2m+1$ 个等距节点 $x_k=a+kh\left(k=0,1,2,\cdots,2m,h=\dfrac{b-a}{2m}\right)$ 处的函数值为 $f(x_k)$ $(k=0,1,2,\cdots,2m)$，且 $f(x)$ 在 $[a,b]$ 上具有连续的四阶导数，则**复合辛普森公式** (9.11) 式是 4 **阶收敛的**，且存在 $\xi\in[a,b]$，使得 (9.11) 式的截断误差为

$$R(f,S_n)=\int_a^b f(x)\,\mathrm{d}x-S_n=-\frac{h^4(b-a)}{180}f^{(4)}(\xi). \tag{9.13}$$

如果记 $M_4=\max\limits_{a\leqslant x\leqslant b}|f^{(4)}(x)|$，则式 (9.11) 在 $[a,b]$ 上的绝对误差限为

$$|R(f,S_n)|=\left|\int_a^b f(x)\,\mathrm{d}x-S_n\right|\leqslant\frac{h^4}{180}M_4(b-a). \tag{9.14}$$

例 9.3.8 估计用辛普森公式计算定积分 $I=\int_0^{\frac{\pi}{2}}\mathrm{e}^{\sin x}\mathrm{d}x$ 时的误差，取 $h=\dfrac{\pi}{40}$.

解 根据估计误差公式 (9.14)，先输入求 $f^{(4)}(x)$ 的程序

```
>>syms x,y=exp(sin(x)); yx4=diff(y,x,4)
```

运行后输出被积函数的四阶导函数（略）. 然后再输入误差估计程序

```
>>h=pi/40; x=0:0.00001:pi/2;
yx4=sin(x).*exp(sin(x))-4*cos(x).^2.*exp(sin(x))+3*sin(x).^2.
*exp(sin(x))-6*sin(x).*cos(x).^2.*exp(sin(x))+cos(x).^4.*exp(sin(x));
juyx4 = abs(yx4); RS=(h^4)*(pi/2)*max(juyx4)/180
```

运行后屏幕显示误差估计值

```
RS =
    3.610450295892220e-006
```

9.3.4 辛普森数值积分的 MATLAB 程序

用 MATLAB 软件和辛普森公式计算数值积分和估计误差限除了用例9.3.7

和例 9.3.8 自己编写的 MATLAB 计算程序外,MATLAB 系统还提供了用自适应辛普森公式计算数值积分的程序 quad.m,具体的调用格式如下:

调用格式一:quad('fun',a,b)

计算函数 $Y=fun(X)$ 在区间 (a,b) 上的数值积分,自动选择步长,在 MATLAB 6.3 中绝对误差限为 10^{-6},在 MATLAB 5.3 中误差限为 10^{-3},输出数值积分值. 其中函数 $Y=fun(X)$ 是以 fun.m 文件命名的 M 文件函数(或库函数如 'sin','log'),或者是 F = inline('fun')形式,或者数组,函数 $Y=fun(X)$ 将接受向量的自变量,并且返回结果是向量 Y,即在 X 的每个元素处的积分估计值.

调用格式二:quad('fun',a,b,tol)

同上,但指定绝对误差限为 tol,默认值为 10^{-6},在 MATLAB 5.3 中误差限为 10^{-3}.

调用格式三:[Q,FCNT] = quad (...)

输出数值积分值 Q 和函数赋值的编号 **FCNT**.

调用格式四:quad('fun',a,b, tol,TRACE)

输入量非零数 TRACE,则会以动态的形式显示在递归求积分的整个过程的 [fcnt a b-a Q]的值;

调用格式五:quad('fun',a,b, tol,TRACE,P1,P2, ...)

此命令规定对函数 fun (X,P1,P2,...)附加的参数 $P_1,P_2,\cdots$ 直接通过. 传递 tol 或 TRACE 的空矩阵使用默认值.

说明:(1) 上面所有的被积函数 fun 的调用格式,都必须指定被积函数 fun 为嵌入对象 F = inline('fun') ,然后调用 Q = quad(F,a,b)或指定为函数处理,例如,将被积函数 fun 作一个名为 myfun.m 的 M 文件

```
function y = myfun(x)
         y = fun(x)
```

然后调用 Q = quad(@myfun,a,b,...).

(2) 在定义被积函数 fun 中的运算用数组算子 .* , ./ 和 .^ ;

(3) 如果没有给定被积函数 $y=fun$ 的具体表达式,而只给定了积分变量和被积函数的离散的数据 $X=(x_0,x_1,\cdots,x_n)$, $Y=(y_0,y_1,\cdots,y_n)$ 或数表:

x	x_0	x_1	x_2	x_3	$\cdots$	x_n
$y=f(x)$	y_0	y_1	y_2	y_3	$\cdots$	y_n

则常用的处理方法如下:

首先利用插值方法(如分段样条插值,分段低阶的拉格朗日插值等方法)求

出分段插值函数 *fun*,然后用说明(1)中的方法重新指定插值函数 fun.

(4) 在 MATLAB 6.5 中可以计算无界函数的反常积分.

我们还可以根据复合辛普森公式编写名为 comsimpson.m 的计算 $\int_a^b f(x)\mathrm{d}x$ 的数值积分的 MATLAB 主程序.

复合辛普森数值积分的 MATLAB 主程序

输入量:*fun* 是被积函数 $f(x)$, a, b 分别是积分下、上限, n 是小区间的数目.

输出量:y 是利用复合辛普森数值积分公式(9.11)计算 $\int_a^b f(x)\mathrm{d}x$ 的数值积分结果.

```
function y = comsimpson(fun,a,b,n)
  z1 = feval (fun,a) + feval (fun,b);m = n/2;
  h = (b - a)/(2 * m); x = a;
  z2 = 0; z3 = 0; x2 = 0; x3 = 0;
for k = 2:2:2 * m
  x2 - x + k * h; z2 = z2 + 2 * feval (fun,x2);
end
for k = 3:2:2 * m
  x3 = x + k * h; z3 = z3 + 4 * feval (fun,x3);
end
y = (z1 + z2 + z3) * h/3;
```

例 9.3.9 用 comsimpson.m 和 quad.m 分别计算定积分 $I=\frac{1}{\sqrt{2\pi}}\int_0^1 \mathrm{e}^{-\frac{x^2}{2}}\mathrm{d}x$, 取精度为 10^{-4},并与精确值比较.

解 (1) 建立 fun.m 文件命名的 M 文件函数

```
function y = fun(x)
          y = exp( ( -x.^2)./2)./(sqrt(2 * pi));
```

(2) 输入程序

```
 > > [Q1,FCNT14] = quad(@ fun,0,1,1.e -4,3),
Q2 = comsimpson (@ fun,0,1,10000)
syms x
fi = int(exp( ( -x.^2)./2)./(sqrt(2 * pi)),x,0, 1); F s = double (fi)
wQ1 = double (abs(fi - Q1) ), wQ2 = double (abs(fi  Q2) )
```

运行后屏幕显示 I 的精确值 Fs,用 comsimpson.m 和 quad.m 分别计算 I 的近似值 Q_2, Q_1 和迭代次数 $FCNT14$,取精度分别为 10^{-4}, Q_2, Q_1 分别与精确值 Fs 的绝对误差 wQ_2, wQ_1 如下

```
      9      0.0000000000     2.71580000e-001     0.1070275100
     11      0.2715800000     4.56840000e-001     0.1597942242
     13      0.7284200000     2.71580000e-001     0.0745230082
Q1 =             FCNT14 =             Q2 =
     0.3413            13                  0.3413
Fs =             wQ1 =                wQ2 =
     0.3413        3.6619e-009          3.7061e-005
```

例 9.3.10 用辛普森公式和梯形公式分别计算定积分 $I=-\int_0^1 e^{x^3}\cos 2x\mathrm{d}x$，取精度分别为 10^{-11} 和 10^{-7}，并与精确值比较，说明二者公式的计算速度.

解 输入程序

```
>> F = inline('-exp(x.^3).*cos(2*x)');
[Q11,FCNT11] = quad(F,0,1,1.e-11)
[Q7,FCNT7] = quad(F,0,1,1.e-7),
h1 = 1/1000000; h2 = 1/10000; x1 =0:h1:1;x2 =0:h2:1;
y1 = -exp(x1.^3).*cos(2*x1);
y2 = -exp(x2.^3).*cos(2*x2); z11c =cumtrapz(x1,y1),
z7c =cumtrapz(x2,y2),
syms x
fi =int( -exp(x.^3).*cos(2*x),x,0, 1); F s = double (fi)
wz11c =abs(F s - z11c),wz7c =abs(F s - z7c),
wQ11 =abs(F s - Q11), wQ7 =abs(F s - Q7),
```

运行后屏幕显示 I 精确值 Fs，用辛普森公式和梯形公式分别计算 I 的近似值 $Q_{11}, Q_7, z_{11c}, z_{7c}$ 和迭代次数 $FCNT_{11}, FCNT_7$，取精度分别为 10^{-11} 和 10^{-7}，并与精确值 Fs 的绝对误差 $wz_{11c}, wz_{7c}, wQ_{11}, wQ_7$ 如下

```
Q11 =                          FCNT11 =
   -0.42854543302034                 305
Q7 =                           FCNT7 =
   -0.42854543101012                 41
z11c =
      ............................
      Columns 999997 through 1000000
 -0.42854995777048   -0.42854882659527   -0.42854769541173
 -0.42854656421986
Column 1000001
 -0.42854543301964
 z7c =
    ..............................................
```

```
 Columns 9997 through 10000
-0.42899724132379   -0.42888441241979   -0.42877150027131
-0.42865850483633
Column 10001
-0.42854542607276
 Warning: Explicit integral could not be found.
 > In C:\MATLAB6p5p1\toolbox\symbolic\@sym\int.m at line 58
 Fs =
      -0.42854543302032
 wz11c =
     .................................
     Columns 999997 through 1000000
0.00000452475016   0.00000339357496   0.00000226239142
0.00000113119954
Column 1000001
0.00000000000068
wz7c =
 .........................................
 Columns 9997 through 10000
0.00045180830347   0.00033897939947   0.00022606725100
0.00011307181601
 Column 10001
 0.00000000694756
wQ11 =                              wQ7 =
  2.337019466835955e-014              2.010192234891406e-009
```

可以看出,辛普森公式的计算速度明显快于梯形公式.如上面输出的结果所示,若要求 $\varepsilon=10^{-11}$,则对于辛普森公式,迭代 $n=305$ 次,即可终止计算,且实际数值积分值 $Q_{11}=-0.428\ 545\ 433\ 020\ 34\approx-0.428\ 545\ 433\ 020\ 3$ 与精确值 $Fs=-0.428\ 545\ 433\ 020\ 32$ 的绝对误差已经达到 $wQ_{11}\approx2.3\times10^{-14}$;而用梯形公式计算,迭代 $n=1\ 000\ 001$ 次才达到要求,且实际数值积分值 $z_{11c}=-0.428\ 545\ 433\ 019\ 64\approx-0.428\ 545\ 433$ 与精确值 Fs 的绝对误差才达到 $wz_{11c}\approx6.8\times10^{-13}<wQ_{11}$. 若要求 $\varepsilon=10^{-7}$,则对于辛普森公式,迭代 $n=41$ 次,即可终止计算,且实际数值积分值 $Q_7=-0.428\ 545\ 431\ 010\ 12\approx-0.428\ 545\ 43$ 与精确值 Fs 的绝对误差已经达到 $wQ_7\approx2.0\times10^{-9}$;而用梯形公式计算,迭代 $n=10\ 001$ 次才达到要求,且实际数值积分值 $z_{7c}=-0.428\ 545\ 426\ 072\ 76\approx-0.428\ 545\ 43$ 与精确值 Fs 的绝对误差才达到 $wz_{11c}\approx6.9\times10^{-9}<wQ_7$. 综上所述,辛普森公式比梯形公式计算定积分的计算速度快且精度高.

例 9.3.11 用 MATLAB 函数 int 和 quad 计算无界函数的反常积分

$$\int_0^1 \frac{1}{\sqrt{x}}\mathrm{d}x.$$

解 方法 1 首先将下面名为 fun1.m 的程序保存为 M 文件

```
function  y = fun1(x)
          y =x.^( -1/2);
```

然后在 MATLAB 工作窗口输入程序

```
>> [Q ,FCNT] = quad(@ fun1,0,1,1.e-14,3)
```

运行后屏幕以滚动形式显示在递归求积分的整个过程的值如下

```
..............................................
   6142    0.9830262500    1.69737500e-002    0.0170463949
   6144    0.9830262500    8.48687500e-003    0.0085414363
   6146    0.9915131250    8.48687500e-003    0.0085049586
Warning: Minimum step size reached; singularity possible.
(Type "warning off MATLAB:quad:MinStepSize" to suppress this
warning.)
> In C:\MATLAB6p5p1\toolbox\matlab\funfun\quad.m at line 85
Q =                        FCNT =
   1.99999999979330              6146
```

方法 2 在 MATLAB 工作窗口输入程序

```
>>  syms x
F =int(x.^( -1/2),x,0,1)
```

运行后屏幕显示求积分的结果

```
F =
   2
```

9.3.5 牛顿-科茨(Newton-Cotes)公式及其误差分析

在构造矩形公式、梯形公式和辛普森公式时,我们使用简单的多项式 $P(x)$ 代替被积函数 $f(x)$ 来构造求积公式. 通常,我们也是用简单的、便于积分的、又逼近于被积函数 $f(x)$ 的函数 $P(x)$ 代替 $f(x)$ 来构造求积公式. 由于多项式不但计算方便,而且容易积分,因此,常取 $P(x)$ 为一个多项式.

设 n 次拉格朗日多项式为

$$L_n(x) = \sum_{k=0}^{n} f(x_k)\cdot l_k(x), \tag{9.15}$$

而 n 次多项式为

$$l_k(x) = \frac{\omega_n(x)}{(x-x_k)\omega'_n(x_k)}, \quad k=0,1,2,\cdots,n, \tag{9.16a}$$

其中 $\omega_n(x)=(x-x_0)(x-x_1)(x-x_2)\cdots(x-x_n)$.

另外,式(9.16a)还可以表示为

$$l_k(x)=\prod_{\substack{k=0\\j\neq k}}^{n}\frac{x-x_k}{x_j-x_k},\ k=0,1,2,\cdots,n. \tag{9.16b}$$

于是

$$\int_a^b f(x)\,\mathrm{d}x=\int_a^b L_n(x)\,\mathrm{d}x+R(f,L_n). \tag{9.17}$$

这样得到一个数值插值求积公式

$$\int_a^b L_n(x)\,\mathrm{d}x\approx\sum_{k=0}^{n}f(x_k)\int_a^b l_k(x)\,\mathrm{d}x=\sum_{k=0}^{n}A_k f(x_k),$$

其中 $A_k=\int_a^b l_k(x)\,\mathrm{d}x$. 将这个结论用定理表述如下.

定理 9.3(n 次拉格朗日插值求积公式) 设函数 $f(x)$ 在闭区间 $[a,b]$ 上的 $n+1$ 个节点 $a=x_0<x_1<x_2<\cdots<x_n=b$ 处有定义,且函数值为 $f(x_k)$ $(k=0,1,2,\cdots,n)$,则在闭区间 $[a,b]$ 上存在

$$\int_a^b L_n(x)\,\mathrm{d}x=I_n(f)=\sum_{k=0}^{n}f(x_k)\int_a^b l_k(x)\,\mathrm{d}x, \tag{9.18}$$

使得(9.17)式成立. 其中 $l_k(x)$ 为(9.16a)式或(9.16b)式,$R(f,I_n)$ 是 $I_n(f)$ 的截断误差,即余项,(9.18)式和(9.16a)式是 n **次拉格朗日插值求积公式**.

推论 9.3(n 次拉格朗日插值求积公式的误差分析) 设函数 $f(x)$ 在闭区间 $[a,b]$ 上的 $n+1$ 个节点 $a=x_0<x_1<x_2<\cdots<x_n=b$ 处有定义,且其函数值为 $f(x_k)$ $(k=0,1,2,\cdots,n)$,$f(x)$ 在 $[a,b]$ 上具有连续的 $n+1$ 阶导数,则存在 $\xi\in[a,b]$,使得 n 次拉格朗日插值求积公式(9.18)式的截断误差 $R(f,I_n)$ 为

$$\begin{aligned}R(f,I_n)&=\int_a^b f(x)\,\mathrm{d}x-I_n(f)\\&=\int_a^b\frac{\omega_n(x)}{(n+1)!}f^{(n+1)}(\xi)\,\mathrm{d}x,\end{aligned} \tag{9.19}$$

如果记 $M_{n+1}=\max\limits_{a\le x\le b}|f^{(n+1)}(x)|$,则在 $[a,b]$ 上(9.18)式的绝对误差限为

$$\begin{aligned}|R(f,I_n)|&=\left|\int_a^b f(x)\,\mathrm{d}x-I_n(f)\right|\\&\le M_{n+1}\int_a^b\frac{|\omega_n(x)|}{(n+1)!}\mathrm{d}x.\end{aligned} \tag{9.20}$$

如果闭区间 $[a,b]$ 上的 $n+1$ 个节点 $a=x_0<x_1<x_2<\cdots<x_n=b$ 是等距节点,即步长 $h=x_{k+1}-x_k=\dfrac{b-a}{n}$ $(k=0,1,2,\cdots,n)$,则由定理 9.3 和推论 9.3 可以得到下面的两个推论.

推论 9.4(n 阶牛顿－科茨公式) 设函数 $f(x)$ 在闭区间 $[a,b]$ 上的 $n+1$ 个等距节点 $x_k=a+kh\ \left(k=0,1,2,\cdots,n,h=\dfrac{b-a}{n}\right)$ 处有定义，且其函数值为 $f(x_k)$ $(k=0,1,2,\cdots,n)$，则在闭区间 $[a,b]$ 上存在 **n 阶牛顿－科茨公式**

$$NC_n(f)=(b-a)\sum_{k=0}^{n}C_k^{(n)}f(x_k), \tag{9.21}$$

使得

$$\int_a^b f(x)\,\mathrm{d}x=NC_n(x)+R(f,NC_n(f)), \tag{9.22}$$

其中**科茨系数** $C_k^{(n)}(k=0,1,2,\cdots,n)$ 为

$$C_k^{(n)}=\frac{(-1)^{n-k}}{n\cdot k!\ \cdot(n-k)!}\int_0^n t(t-1)\cdots(t-k+1)(t-k-1)\cdots(t-n)\,\mathrm{d}t. \tag{9.23}$$

从 n 阶科茨系数(9.23)式可以看出，科茨系数只与 n 有关，与积分区间和被积函数无关. 因此，只要给出 n，就可以算出科茨系数，从而写出 n 阶牛顿－科茨公式. 应用 n 阶牛顿－科茨公式(9.21)计算定积分 $\int_a^b f(x)\,\mathrm{d}x$ 时，一方面由于它是由(9. 22)式去掉余项 $R(f,NC_n(f))$ 得到的，因而产生截断误差 $R(f,NC_n(f))$；另一方面，由于计算机的字长是有限的，函数值可能带有误差，并且计算 $NC_n(f)$ 的过程中还会有舍入误差. 关于截断误差 $R(f,NC_n(f))$ 有下面的推论.

推论 9.5(n 阶牛顿－科茨公式的误差分析) (1) 当 n 为偶数时，设函数 $f(x)$ 在闭区间 $[a,b]$ 上具有 $n+2$ 阶连续导数，则存在 $\xi\in[a,b]$，使得 n 阶牛顿－科茨公式(9.21)的截断误差 $R(f,NC_n(f))$ 为

$$R(f,NC_n(f))=\frac{h^{n+3}f^{(n+2)}(\xi)}{(n+2)!}\int_0^n t^2(t-1)(t-2)(t-3)\cdots(t-n)\,\mathrm{d}t. \tag{9.24}$$

(2) 当 n 为奇数时，设函数 $f(x)$ 在闭区间 $[a,b]$ 上具有 $n+1$ 阶连续导数，则存在 $\xi\in[a,b]$，使得 n 阶牛顿－科茨公式(9.21)的截断误差 $R(f,NC_n(f))$ 为

$$R(f,NC_n(f))=\frac{h^{n+2}f^{(n+1)}(\xi)}{(n+1)!}\int_0^n t(t-1)(t-2)(t-3)\cdots(t-n)\,\mathrm{d}t. \tag{9.25}$$

例如，当 $n=4$ 时，得到**基本科茨公式**

$$\int_a^b f(x)\,\mathrm{d}x\approx\frac{b-a}{90}\left[7f(a)+32f\left(a+\frac{b-a}{4}\right)+12f\left(a+2\cdot\frac{b-a}{4}\right)+32f\left(a+3\cdot\frac{b-a}{4}\right)+7f(b)\right]. \tag{9.26}$$

基本科茨公式(9.26)的截断误差 $R(f,NC_6(f))$ 为

$$R(f,NC_6(f))=-\frac{8}{945}\left(\frac{b-a}{4}\right)^7 f^{(6)}(\xi),\xi\in[a,b]. \tag{9.27}$$

下面考虑 n 阶牛顿－科茨公式与梯形公式、辛普森公式的关系.

(1) 当 $n=1$ 时,得到基本梯形公式

$$\int_a^b f(x)\,dx\approx\frac{b-a}{2}[f(a)+f(b)], \tag{9.28}$$

截断误差为

$$R(f,NC_1(x))=-\frac{(b-a)^3}{12}f''(\xi),\xi\in[a,b]. \tag{9.29}$$

(2) 当 $n=2$ 时,得到基本辛普森公式

$$\int_a^b f(x)\,dx\approx\frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right], \tag{9.30}$$

截断误差为

$$R(f,NC_2(x))=-\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi),\xi\in[a,b]. \tag{9.31}$$

(3) 当 $n=3$ 时,得到 3/8 辛普森公式

$$\int_a^b f(x)\,dx\approx\frac{b-a}{8}\left[f(a)+3f\left(a+\frac{b-a}{3}\right)+3f\left(a+2\cdot\frac{b-a}{3}\right)+f(b)\right]. \tag{9.32}$$

为了便于应用,我们将 $n=1,2,\cdots,8$ 的科茨系数 $C_k^{(n)}(k=0,1,2,\cdots,n)$ 列入表 9－2,利用这张表,可以很快地写出 $n=1,2,\cdots,8$ 的牛顿－科茨公式.

表 9－2 $n=1,2,\cdots,8$ 的科茨系数 $C_k^{(n)}(k=0,1,2,\cdots,n)$

n	$n=1,2,\cdots,8$ 的科茨系数 $C_k^{(n)}(k=0,1,2,\cdots,n)$						
1	$\frac{1}{2}$	$\frac{1}{2}$					
2	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$				
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$			
4	$\frac{7}{90}$	$\frac{16}{45}$	$\frac{2}{15}$	$\frac{16}{45}$	$\frac{7}{90}$		
5	$\frac{19}{288}$	$\frac{25}{96}$	$\frac{25}{144}$	$\frac{25}{144}$	$\frac{25}{96}$	$\frac{19}{288}$	
6	$\frac{41}{840}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{34}{105}$	$\frac{9}{280}$	$\frac{9}{35}$	$\frac{41}{840}$

续表

n	$n=1,2,\cdots,8$ 的科茨系数 $C_k^{(n)}(k=0,1,2,\cdots,n)$								
7	$\frac{751}{17\ 280}$	$\frac{3\ 577}{17\ 280}$	$\frac{1\ 323}{17\ 280}$	$\frac{2\ 989}{17\ 280}$	$\frac{2\ 989}{17\ 280}$	$\frac{1\ 323}{17\ 280}$	$\frac{3\ 577}{17\ 280}$	$\frac{751}{17\ 280}$	
8	$\frac{989}{28\ 350}$	$\frac{5\ 888}{28\ 350}$	$-\frac{928}{28\ 350}$	$\frac{10\ 496}{28\ 350}$	$-\frac{4\ 540}{28\ 350}$	$\frac{10\ 496}{28\ 350}$	$-\frac{928}{28\ 350}$	$\frac{5\ 888}{28\ 350}$	$\frac{989}{28\ 350}$

例 9.3.12 写出 8 阶牛顿－科茨公式,并用 MATLAB 软件求用该公式计算定积分 $I=\int_a^b e^{\sin x}dx$ 的误差公式.

解 (1) 从表 9－2 可以查出 8 阶牛顿－科茨公式的科茨系数 $C_k^{(n)}$ $(k=0,1,2,\cdots,8)$ 为

$$C_0^{(8)}=C_8^{(8)}=\frac{989}{28\ 350},\ C_1^{(8)}=C_7^{(8)}=\frac{5\ 888}{28\ 350},$$

$$C_2^{(8)}=C_6^{(8)}=-\frac{928}{28\ 350},\ C_3^{(8)}=C_5^{(8)}=\frac{10\ 496}{28\ 350},$$

$$C_4^{(8)}=-\frac{4\ 540}{28\ 350},\ x_k=a+k\frac{b-a}{8}\ (k=0,1,2,3,4,5,6,7,8),$$

根据牛顿－科茨公式(9.21)得 8 阶牛顿－科茨公式为

$$\int_a^b f(x)dx\approx(b-a)\sum_{k=0}^{8}C_k^{(8)}f(x_k),$$

即
$$NC_8(f)=(b-a)\left\{\frac{989}{28\ 350}[f(x_0)+f(x_8)]+\frac{5\ 888}{28\ 350}[f(x_1)+f(x_7)]-\right.$$
$$\left.\frac{928}{28\ 350}[f(x_2)+f(x_6)]+\frac{10\ 496}{28\ 350}[f(x_3)+f(x_5)]-\frac{4\ 540}{28\ 350}f(x_4)\right\}. \tag{9.33}$$

(2) 根据牛顿－科茨公式的截断误差公式(9.24),即

$$R(f,NC_n(f))=\frac{h^{n+3}f^{(n+2)}(\xi)}{(n+2)!}\int_0^n t^2(t-1)(t-2)(t-3)\cdots(t-n)dt,$$

输入程序

```
>> su =1;
for k =1:10,
  su = su * k;
end
  su
  syms t a b M
  f =t * t * (t -1) * (t  2) * (t -3) * (t -4) * (t -5) * (t -6) *
(t -7) * (t -8);
```

```
intf = int(f,t,0,8), y = double(intf),
RNC8 = (((b - a)/8)^11) * M * y/su
```

运行后屏幕显示结果如下

```
su =            intf =              y =
  3628800         -606208/33          -1.836993939393939e+004
RNC8 =
  -35065906189543/6926923254988800 * (1/8 * b - 1/8 * a)^11 * M
```

即,8 阶牛顿-科茨公式的截断误差公式为

$$R(f,NC_8(f)) = -\frac{50}{9\,877}\left(\frac{b-a}{8}\right)^{11} f^{(10)}(\xi),\ \xi \in [a,b]. \qquad (9.34)$$

例 9.3.13 用 MATLAB 软件估计用 8 阶牛顿-科茨公式计算定积分 $I = \int_0^{\frac{\pi}{2}} e^{\sin x} dx$ 的误差,取 15 位小数.

解 下面分别用(9.34)式和牛顿-科茨公式的截断误差公式(9.24)两种方法估计误差.

先输入求 $f^{(10)}(x)$ 的程序

```
>> syms x
y = exp(sin(x)); yx10 = diff(y,x,10)
```

运行后输出被积函数的十阶导函数(略),然后再根据(9.24)式和(9.34)式,用两种方法估计误差,输入误差估计程序

```
>>  syms t
a = 0;b = pi/2; f = t * t * (t - 1) * (t - 2) * (t - 3) * (t - 4) * (t - 5) *
(t - 6) * (t - 7) * (t - 8);
intf = int(f,t,0,8); y = double(intf); h = pi/40; x = 0:0.00001:pi/2;
yx10 = - 120 * cos(x).^8.* exp(sin(x)) + 256 * cos(x).^2.*
exp(sin(x)) + 2352 * cos(x).^6.* exp(sin(x)) - 5440 * cos(x).^4.*
exp(sin(x)) + cos(x).^10.* exp(sin(x)) - 45 * cos(x).^8.* exp(sin(x)).*
sin(x) - 3150 * sin(x).^3.* cos(x).^4.* exp(sin(x)) + 630 * sin(x).^2.* cos(x).^
6.* exp(sin(x)) + 4725 * sin(x).^4.* cos(x).^2.* exp(sin(x)) + 2730 *
sin(x).* cos(x).^6.* exp(sin(x)) - 15750 * sin(x).^2.* cos(x).^4.*
exp(sin(x)) + 22050 * sin(x).^3.* cos(x).^2.* exp(sin(x)) - 19530 * sin(x).*
cos(x).^4.* exp(sin(x)) + 25515 * sin(x).^2.* cos(x).^2.* exp(sin(x)) +
7125 * sin(x).* cos(x).^2.* exp(sin(x)) - sin(x).* exp(sin(x)) - 255 *
sin(x).^2.* exp(sin(x)) - 2205 * sin(x).^3.* exp(sin(x)) - 3150 * sin(x).
^4.* exp(sin(x)) - 945 * sin(x).^5.* exp(sin(x));
myx10 = max(yx10), su = 1;
for k = 1:10
```

```
        su = su * k;
        end
        su; RNC8 = abs((((b - a) / 8)^11) * myx10 * y / su)
         sRNC8 = abs((((b - a) / 8)^11) * myx10 * 35065906189543 /
6926923254988800)
```

运行后屏幕显示误差估计值如下

```
myx10 =                      RNC8 =
   1.307200234011939e+004       1.106663529789837e-006
sRNC8 =
   1.106663529789837e-006
```

由此可见,取 15 位小数时,用两种方法估计 8 阶牛顿-科茨公式计算定积分 I 的误差都为 $1.106\,663\,529\,789\,84\times10^{-6}$.

说明:我们可以证明,当科茨系数 $C_k^{(n)}(k=0,1,2,\cdots,n)$ 都为正数时,相应的 8 阶牛顿-科茨公式是稳定的. 但是从表 9-2 可以看出,当 $n\geqslant8$ 时,科茨系数 $C_k^{(n)}(k=0,1,2,\cdots,n)$ 有正数,也有负数,此时该公式的稳定性没有保证. 另外,并非对一切连数函数 $f(x)$,当 $n\to\infty$ 时,截断误差 $E(f,NC_n(f))\to0$,即牛顿-科茨公式的收敛性没有保证. 所以在实际计算时很少用高阶的牛顿-科茨公式,而是通过低阶复合途径求数值积分,也就是将区间 $[a,b]$ 分成 n 个等长的小区间 $[x_{k-1},x_k]$,在每个小区间 $[x_{k-1},x_k]$ 上用低阶的牛顿-科茨公式计算数值积分,然后累加这些近似值得到所求的积分近似值,这就是我们在前面介绍过的**复合法**,用此法得到的公式称为**复合求积公式**.

如果将区间 $[a,b]$ 分成 n 等份,则步长 $h=\dfrac{b-a}{n}$, $x_k=a+(k-1)h$ $(k=0,1,2,\cdots,n)$ 可以得到前面介绍的复合梯形求积公式和复合辛普森公式以及下面的复合科茨求积公式.

推论 9.6(复合科茨公式及其误差分析) 如果将区间 $[a,b]$ 分成 n 等份,则步长 $h=\dfrac{b-a}{n}$, $x_k=a+(k-1)h$ $(k=0,1,2,\cdots,n)$,则有**复合科茨求积公式**

$$\int_a^b f(x)\,dx\approx\frac{h}{90}\left[7f(a)+32\sum_{k=0}^{n-1}f\left(x_k+\frac{h}{4}\right)+12\sum_{k=0}^{n-1}f\left(x_k+\frac{h}{2}\right)+\right.$$
$$\left.32\sum_{k=0}^{n-1}f\left(x_k+\frac{3h}{4}\right)+14\sum_{k=0}^{n-1}f(x_k)+7f(b)\right].\qquad(9.35)$$

如果函数 $f(x)$ 在闭区间 $[a,b]$ 上具有连续的 6 阶导数,则存在 $\xi\in[a,b]$ 和 $\xi_k\in[x_k,x_{k+1}]$ $(k=0,1,2,\cdots,n-1)$,使得(9.35)式在 $[a,b]$ 上的截断误差 $R(f,NC(f))$ 为

$$R(f,NC(f))=-\sum_{k=0}^{n-1}\frac{8}{945}\left(\frac{b-a}{4n}\right)^7f^{(6)}(\xi_k),\qquad(9.36a)$$

或

$$R(f,NC(f)) = -\frac{8n}{945}\left(\frac{b-a}{4n}\right)^7 f^{(6)}(\xi). \tag{9.36b}$$

9.3.6 牛顿－科茨数值积分和误差分析的 MATLAB 程序

下面介绍计算科茨系数 $C_k^{(n)}$ 和估计(9.21)式在 $[a,b]$ 上的截断误差 $R(f,NC(f))$ 的程序及其用法，并且介绍根据在 MATLAB 中提供了用自适应递归 8 阶牛顿－科茨公式计算数值积分的程序 quad8.m 和使用方法.

(一) 估计误差的 MATLAB 程序

根据计算定积分 $\int_a^b f(x)\mathrm{d}x$ 的近似值的 n 阶牛顿－科茨公式的截断误差公式(9.24)式和(9.25)式编写的求(9.21)式在 $[a,b]$ 上的截断误差 $R(f,NC(f))$ 公式的主程序如下：

> **计算 n 阶牛顿－科茨的公式的截断误差公式的 MATLAB 主程序**
> **输入量**：牛顿－科茨的公式(9.21)的阶数 n.
> **输出量**：RNC 是对应的 n 阶牛顿－科茨的公式(9.21)的截断误差 $R(f,NC(f))$ 公式.

```
function RNC = ncE(n)
suk = 1; p = n/2 - fix(n/2);
if p = = 0
  for k = 1:n + 2
    suk = suk * k;
    end
    suk; syms t a b fxn2,su = t^2;
    for u = 1:n
      su = su * (t - u);
    end
    su; intf = int(su,t,0,n); y = double(intf);
    RNC = (((b - a)/n)^(n + 3)) * fxn2 * abs(y)/suk;
  else
    for k = 1:n + 1
      suk = suk * k;
    end
      suk; syms t a b fxn1,su = t;
    for u = 1:n
      su = su * (t - u);
```

```
      end
        su; intf = int(su,t,0,n); y = double(intf);
        RNC = (((b - a)/n)^(n + 2)) * fxn1 * abs(y)/suk;
    end
```

例 9.3.14 用求截断误差公式的 MATLAB 主程序,求计算定积分 $\int_a^b f(x)\,dx$ 的近似值的 1,2,3,4,8 阶牛顿 – 科茨公式的截断误差公式.

解 输入求 1,2,3,4,8 阶牛顿 – 科茨公式的截断误差公式的程序

```
>> n = 1, RNC1 = ncE(n), n = 2, RNC2 = ncE(n), n = 3, RNC3 = ncE(n)
n = 4, RNC4 = ncE(n), n = 8, RNC8 = ncE(n)
```

运行后屏幕显示结果如下

```
n =              RNC1 =
    1                 1/12 * (b - a)^3 * fxn1
n =              RNC2 =
    2                 1/90 * (1/2 * b - 1/2 * a)^5 * fxn2
n =              RNC3 =
    3                 3/80 * (1/3 * b - 1/3 * a)^5 * fxn1
n =              RNC4 =
    4                 8/945 * (1/4 * b - 1/4 * a)^7 * fxn2
n =              RNC8 -
    8            35065906189543/6926923254988800 * (1/8 * b - 1/8 *
a)^11 * fxn2
```

例 9.3.15 用 MATLAB 软件估计用 5,6 阶牛顿 – 科茨公式计算定积分 $I = \int_0^{\frac{\pi}{2}} e^{\sin x}\,dx$ 的截断误差公式和误差限,取 15 位小数.

解 输入求 $n = 5,6$ 阶牛顿 – 科茨公式的截断误差公式和被积函数的 6,8 阶导函数的程序

```
>> n = 5;RNC5 = ncE(n), n = 6;RNC6 = ncE(n)
syms x
y = exp(sin(x)); yx6 = diff(y,x,6), yx8 = diff(y,x,8)
```

运行后输出被积函数的 6,8 阶导函数(略)和 $n = 5,6$ 阶牛顿 – 科茨公式的截断误差公式如下

```
RNC5 =                                      RNC6 =
    275/12096 * (1/5 * b - 1/5 * a)^7 * fxn1   9/1400 * (1/6 * b - 1/6 * a)
^9 * fxn2
```

然后再输入误差估计程序

```
>> a = 0;b = pi/2; h = pi/40; x = 0:0.00001:pi/2;
yx6 = - sin(x).* exp(sin(x)) + 16 * cos(x).^2.* exp(sin(x)) - 15
```

```
* sin(x).^2.* exp(sin(x)) +75 * sin(x).* cos(x).^2.* exp(sin(x)) -20 *
cos(x).^4.* exp(sin(x)) -15 * sin(x).^3.* exp(sin(x)) +45 * sin(x).^2.*
cos(x).^2.* exp(sin(x)) -15 * sin(x).* cos(x).^4.* exp(sin(x)) +
cos(x).^6.* exp(sin(x));
        yx8 = cos(x).^8.* exp(sin(x)) -756 * sin(x).* cos(x).^2.*
exp(sin(x)) -1260 * sin(x).^2.* cos(x).^2.* exp(sin(x)) +630 * sin(x).*
cos(x).^4.* exp(sin(x)) -420 * sin(x).^3.* cos(x).^2.* exp(sin(x)) +210 *
sin(x).^2.* cos(x).^4.* exp(sin(x)) -28 * sin(x).* cos(x).^6.* exp(sin(x)) -
56 * cos(x).^6.* exp(sin(x)) + sin(x).* exp(sin(x)) +63 * sin(x).^2.*
exp(sin(x)) +210 * sin(x).^3.* exp(sin(x)) +105 * sin(x).^4.* exp(sin(x)) -
64 * cos(x).^2.* exp(sin(x)) +336 * cos(x).^4.* exp(sin(x));
        myx6 = max(yx6); myx8 = max(yx8); RNC5 =275/12096 * (1/5 * b -
1/5 * a)^7 * myx6
        RNC6 -9/1400 * (1/6 * b -1/6 * a)^9 * myx8
```

运行后屏幕显示误差限如下

```
        RNC5 =                              RNC6 =
         3.625385713339320e-004              3.826183594914823e-005
```

(二) 计算科茨系数 $C_k^{(n)}$ 和求截断误差公式的 MATLAB 程序

根据计算定积分 $I=\int_a^b f(x)\mathrm{d}x$ 的 n 阶牛顿-科茨公式(9.21),(9.23)编写的计算科茨系数 $C_k^{(n)}(k=0,1,2,\cdots,n)$ 主程序如下：

> **计算 n 阶科茨系数 $C_k^{(n)}$ 和求截断误差公式的 MATLAB 主程序**
>
> 输入量：牛顿-科茨的公式(9.21)的阶数 n.
>
> 输出量：*Cn* 是计算定积分 I 的 n 阶牛顿-科茨公式的科茨系数 $C_k^{(n)}(k=0,1,2,\cdots,n)$，*RNCn* 是对应的 n 阶牛顿-科茨公式(9.21)的截断误差 $R(f,NC(f))$ 公式.

```
        function [Cn, RNCn] = newcotE(n)
        syms t a b M,Fz = zeros(1,n +1);
        Cn = zeros(1,n +1); su = t;k =1;m =1;m0 =1;
          for u =1:n
            su1 = su * (t -u);m01 = m0 * u; su = su1;m0 = m01;
          end
            su;m0; f1 = su/(t -0); intf1 = int(f1,t,0,n);
            y = double(intf1);
            Cn(1) = (( -1)^(n -0) * y)/(n * m0); k =1;m =1;
          for j =1:n
            k1 = k * j; m1 = m * (n -j); f = su/(t -j);
```

```
    intf = int(f,t,0,n); y = double(intf);
    Cn(j +1) = (( -1)^(n - j) * y)/(n * k1 * m1);
    warning off MATLAB:divideByZero
  end
     fn = su/(t - n); intfn = int(fn,t,0,n);
      y = double(intfn);Cn(n +1) = y/(n * m0);
      Cn; suk = 1; p = n/2 - fix(n/2);
  if p = = 0
    for k = 1:n +2
      suk = suk * k;
    end
      suk; syms t a b fxn2,su = t^2;
    for u = 1:n
       su = su * (t - u);
    end
      su; intf = int(su,t,0,n); y = double(intf);
      RNCn = (((b - a)/n)^(n +3)) * fxn2 * abs(y)/suk;
 else
  for k = 1:n +1
    suk = suk * k;
  end
    suk; syms t a b fxn1,su = t;
   for u = 1:n
       su = su * (t - u);
   end
   su; intf = int(su,t,0,n); y = double(intf);
   RNCn = (((b - a)/n)^(n +2)) * fxn1 * abs(y)/suk;
 end
```

例 9.3.16 用计算 n 阶科茨系数 $C_k^{(n)}$ 和求截断误差公式的 MATLAB 主程序,计算定积分 $I=\int_a^b f(x)\,\mathrm{d}x$ 的 1 ~3 阶牛顿 - 科茨公式的系数和截断误差公式.

解 先求 1 ~3 阶牛顿 - 科茨公式的系数和截断误差公式.输入程序

```
>>  n1 = 1,[Cn1,RNCn1] = newcotE(n1)
n2 = 2,[Cn2,RNCn2] = newcotE(n2)
n3 = 3,[Cn3,RNCn3] = newcotE(n3)
```

运行后屏幕显示 1 ~3 阶牛顿 - 科茨公式的系数 Cn_1, Cn_2, Cn_3 和截断误差公式 $RNCn_1, RNCn_2, RNCn_3$ 如下

```
n1 =
     1
Cn1 =
    0.50000000000000   0.50000000000000
RNCn1 =
1/12*(b-a)^3*fxn1
n2 =
     2
Cn2 -
    0.16666666666667   0.66666666666667   0.16666666666667
RNCn2 =
   1/90*(1/2*b-1/2*a)^5*fxn2
n3 =
     3
Cn3 =
 0.12500000000000   0.37500000000000   0.37500000000000   0.12500000000000
RNCn3 =
 3/80*(1/3*b-1/3*a)^5*fxn1
```

(三)计算 8 阶牛顿-科茨公式的 MATLAB 程序

在 MATLAB 5.3 中提供了用自适应递归 8 阶牛顿-科茨公式计算数值积分的程序 quad8.m.但是,因为 quad8.m 递归速度慢和其他原因,在 MATLAB 6.5 和 MATLAB 7.01 中极力推荐使用程序 quadl.m 取代 quad8.m. 在 MATLAB 7.01 中虽然还保留 quad8.m 的文件名,实际上已经将它的程序废弃,输入有关 quad8 的程序时,运行的是 quadl.m 的程序.quad8.m 和 quadl.m 的调用方法与 quad 类似,下面介绍 quad8.m 的调用格式.

调用格式一:quad8('fun',a,b)

计算函数 $Y=fun(X)$ 在区间 (a,b) 上的数值积分,自动选择步长,相对误差限为 10^{-3},输出数值积分值. 函数 $Y=fun(X)$ 将接受向量的自变量 X,并且返回结果是向量 Y,即在 X 的每个元处的积分估计值.

调用格式二:quad8('fun',a,b,tol)

同上,返回相对误差 tol 的数值积分,如果 TOL = [rel_tol abs_tol],则 rel tol 和 abs_tol 分别表示相对误差和绝对误差.

调用格式三:quad8('fun',a,b, tol,TRACE)

输入非零数 TRACE,则会以动态的形式显示误差为 tol 的数值积分在递归求积分的整个过程和图形.

调用格式四:quad8('fun',a,b, tol,TRACE,P1,P2,...)

此命令规定对函数 fun(X,P1,P2,...)附加的参数 $P_1,P_2,\cdots$ 直接通过.传递 tol 或 TRACE 的空矩阵使用默认值.

说明:(1) 上面所有的函数 fun 的调用格式,都必须指定被积函数 fun 为嵌入对象 F = inline('fun'),然后调用 Q = quad8(F,a,b)或指定为函数处理,例如,将被积函数 fun 作一个名为 myfun.m 的 M 文件

```
function y = myfun(x)
         y = fun(x)
```

然后调用 Q = quad8(@ myfun,a,b,...).

(2) 定义被积函数 fun 中的运算使用数组算子 .* , ./ 和 .^ .

(3) 如果没有给定被积函数 y = fun(x)的具体表达式,而只给定了积分变量和被积函数的离散的数据 $\boldsymbol{X}=(x_0,x_1,\cdots,x_n)$,$\boldsymbol{Y}=(y_0,y_1,\cdots,y_n)$,或数表

x	x_0	x_1	x_2	x_3	$\cdots$	x_n
$y=f(x)$	y_0	y_1	y_2	y_3	$\cdots$	y_n

常用的处理方法如下:

首先利用插值方法(如分段样条插值,分段低阶的拉格朗日插值等)求出分段插值函数 *fun*,然后用说明(1)中的方法重新指定插值函数 fun.

例 9.3.17 (1) 用梯形公式、辛普森和 8 阶牛顿-科茨求积公式计算定积分 $I=\int_0^{\frac{\pi}{2}} e^{\sin x}dx$,取精度 10^{-4},作出它们的积分图,并与精确值进行比较;

(2) 分别在 MATLAB 6.5 和 MATLAB 7.01 中用 quad8 计算定积分 I,观察运行结果有何变化,为什么?

解 (1) ① 用梯形求积公式计算定积分. 输入程序

```
>> h = pi/500; x = 0:h:pi/2; y = exp(sin(x));
zt = trapz(x,y), ztc = cumtrapz(x,y), plot(x, ztc,'ro')
```

运行后屏幕显示用函数 trapz 和 cumtrapz 分别计算结果 *zt*,*ztc* 分别如下

```
zt =
     3.10437572798742
ztc =
    Columns 1 through 3
                        0    0.00630298652792    0.01264569951380
.................................................
Columns 250 through 251
    3.08729642810745    3.10437572798742
```

② 用辛普森求积公式计算定积分. 输入程序

```
>> syms x
L = inline('exp(sin(x))');
```

```
[QS,FCNTS] =quad(L,0, pi/2,1.e-4,2)
```

运行后屏幕显示用辛普森求积公式计算定积分的值 *QS* 和递归次数 *FCNTS* 分别如下

```
QS =                          FCNTS =
  3.10438133817254                13
```

③ 用 8 阶牛顿－科茨求积公式计算定积分. 在 MATLAB 6.5 中输入程序

```
>> syms x
L = inline('exp(sin(x))');
[Q8,FCNT8] = quad8(L,0, pi/2,1.e-4,3)
```

运行后屏幕显示用 8 阶牛顿－科茨求积公式计算定积分的值 *Q8* 和递归次数 *FCNTS* 分别如下

```
Q8 =                             FCNT8 =
    3.10437901785555                 33
```

④ 输入求定积分的精确值的程序

```
>> syms x
y=exp(sin(x)); F=int(y,0,pi/2), Fj=double(F)
wzt=abs( Fj - zt), wQS = abs(Fj - QS), wQ8 = abs(Fj - Q8)
```

运行后屏幕显示计算的定积分值 *F* 和近似值 F_j,梯形公式、辛普森和 8 阶牛顿－科茨求积公式计算定积分的值与 F_j的绝对误差 *wztc*,*wQS* 和 wQ_8如下

```
Warning: Explicit integral could not be found.
> In C:\MATLAB6p5p1\toolbox\symbolic\@sym\int.m at line 58
F =
int(exp(sin(x)),x = 0 .. 1/2*pi)
Fj =                              wzt =
    3.10437901785556                 3.289868133471430e-006
wQS -                             wQ8 =
    2.320316987436399e-006           7.993605777301127e-015
```

用 8 阶牛顿－科茨求积公式计算定积分的值,迭代 33 次,计算结果 3.104 379 017 855 55的绝对误差 7.993 6e－015 最小,远远超出了调用 quad8 (L,0, pi/2,1.e-4,3)时给定的精度 10^{-4},说明 quad8 中的限定条件 10^{-4} 起的作用不大,这是 quad8 应该改进的地方;其次,用辛普森求积公式计算定积分的值,迭代 13 次,计算结果 3.104 381 338 172 54 的绝对误差为 2.320 3e－006 小于给定的精度 10^{-4},并且二者很接近,说明 quad 中的限定条件 10^{-4}的作用很大;用梯形公式计算定积分的值,迭代 250 次,计算结果3.104 375 727 987 42 的绝对误差 3.289 9e－006 最大,但是达到了给定的精度 10^{-4}. 由此可见,梯形公式计算定积分的速度比 8 阶牛顿－科茨公式和辛普森公式慢得多. 对于给定的精度 10^{-4},用这三种方法计算定积分的值 $I\approx 3.104\ 4$.

输入画图程序

```
>> syms x
F = int(exp(sin(x)),0,pi/2),Fj = double(F),plot(Fj,'ro'),
hold on
L = inline('exp(sin(x))');
Q = quad(L,0, pi/2,1.e-4,2),plot(Q,'g*')
hold off,hold on, h = pi/40; x = 0:h:pi/2; y - exp(sin(x));
ztc = cumtrapz(x,y), plot(x, ztc,'mp'), hold off
hold on, [Q8,FCNT8] = quad8(L,0, pi/2,1.e-4,3), hold off
grid, xlabel('自变量 X'), ylabel('因变量 Y')
title('8 阶牛顿-科茨公式和梯形公式计算数值积分递归图')
legend('精确值', '辛普森公式计算定积分','梯形公式计算定积分递归图','8
阶牛顿-科茨公式计算定积分递归图')
```

运行后屏幕显示图 9-3.

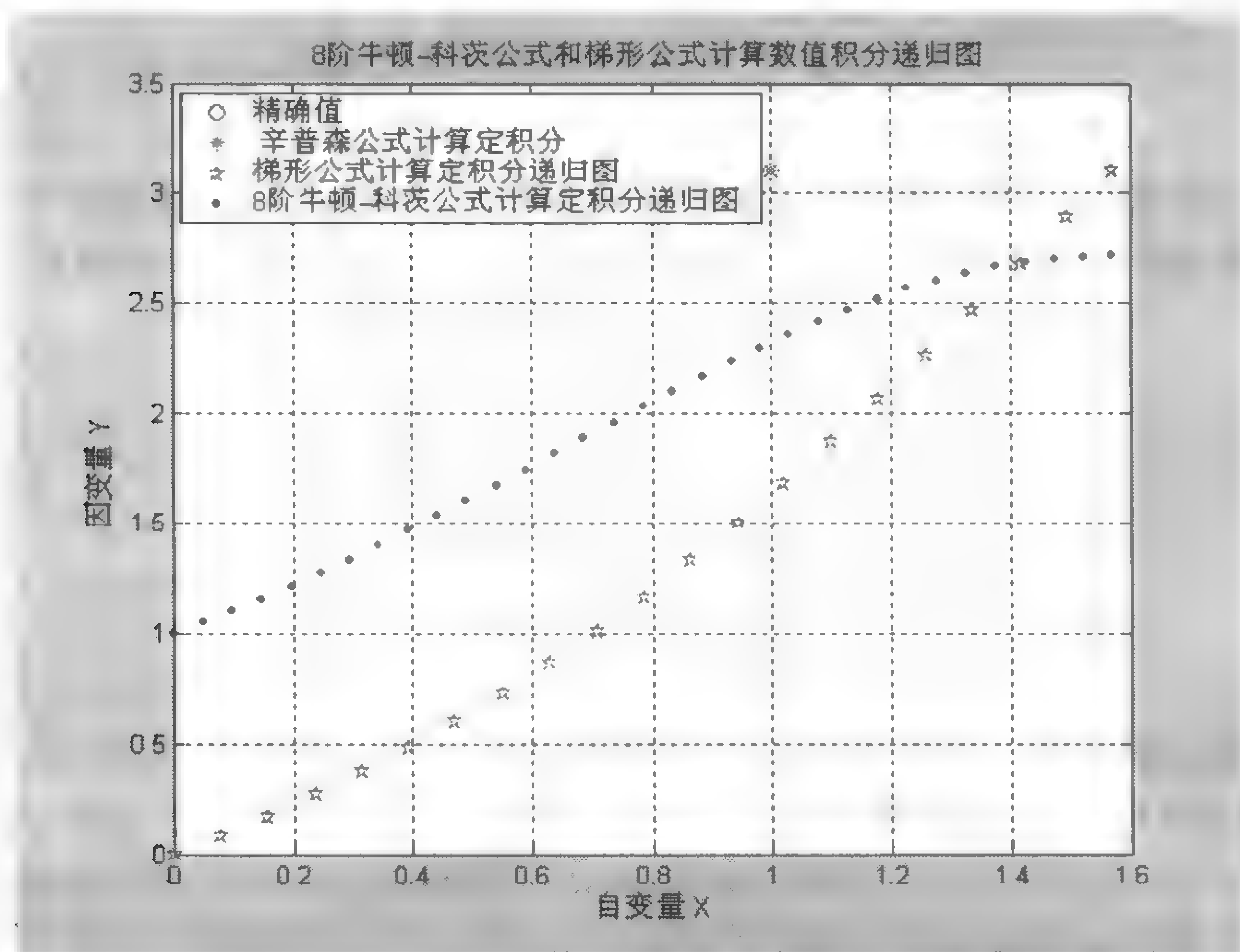


图 9-3 8 阶牛顿-科茨公式和梯形公式计算数值积分递归图

(2) 在 MATLAB 7.01 中输入与(1)③相同的程序,运行后屏幕显示用 8 阶牛顿-科茨求积公式计算定积分的值 *Q8* 和递归次数 *FCNTS* 分别如下

```
18      0.0000000000      7.85398163e-001      3.1043791092
```

```
Q8 =                                    FCNT8 =
        3.10437910915980                     18
```

可见,同一套程序在 MATLAB 6.5 和 MATLAB 7.01 中运行的结果不同. 因为 MATLAB 7.01 实际运行的是 QUADL 程序.

9.3.7 利用三次样条求表格型数值积分的 MATLAB 方法

(一) 利用插值多项式求表格型数值积分的方法

如果没有给出定积分 $I=\int_a^b f(x)\mathrm{d}x$ 中被积函数 $y=f(x)$ 的具体表达式,而只是给定了积分变量 x 和被积函数的离散的数据 $\boldsymbol{X}=(x_0,x_1,\cdots,x_n)$, $\boldsymbol{Y}=(y_0,y_1,\cdots,y_n)$,或数表

x	x_0	x_1	x_2	x_3	$\cdots$	x_n
$y=f(x)$	y_0	y_1	y_2	y_3	$\cdots$	y_n

则利用插值多项式求表格型数值积分常用的处理方法的步骤如下:

步骤 1 首先利用插值方法(如用分段二、三阶拉格朗日插值多项式、牛顿插值、分段埃尔米特插值、样条插值等)构造分段插值函数 $P(x)$ 逼近被积函数 $f(x)$;

步骤 2 然后用下面的方法之一将分段插值函数 $P(x)$ 重新指定为在 MATLAB中可运行的函数 fun:

方法 1 指定被积函数 fun 为嵌入对象 F = inline('fun');然后调用 Q = quad8(F,a,b)或 Q = quad(F,a,b)或 Q = quadl(F,a,b)计算数值积分;

方法 2 将被积函数 fun 指定为函数处理,即建立并保存一个 M 函数文件.

例如,将被积函数 fun 作一个名为 fun1.m 的 M 文件

```
function y = fun1(x)
         y = fun(x)
```

然后调用 Q = quad8(@ fun1,a,b,...)或 Q = quad(@ fun1,a,b,...)或 Q = quadl(@ fun1,a,b,...)计算数值积分.

(二) 利用三次样条求表格型数值积分的方法

根据给定的离散数据 $\boldsymbol{X}=(x_0,x_1,\cdots,x_n)$, $\boldsymbol{Y}=(y_0,y_1,\cdots,y_n)$或数表,利用 MATLAB 库函数中的三次样条函数 spline 构造分段插值函数 y = fun(x),使 y 逼近被积函数 $f(x)$,用数值积分的方法计算定积分 $I=\int_a^b f(x)\mathrm{d}x$,可以在 MATLAB 工作窗口直接输入下面的程序

```
a = a1,b = b1, X = [x0,x1,...,xn];Y = [y0,y1,...,yn],
pp = spline(x,y)
[Q,FCNT] = quad(@ ppval,a1,b1,[],[],pp)
或[Q1,FCNT1] = quadl(@ ppval,a1,b1,[],[],pp)
或 [Q8,FCNT8] = quad8(@ ppval,a1,b1,[],[],pp)
```

运行后输出的结果即为所求.

例 9.3.18 给出概率积分 $f(x)=\frac{1}{\sqrt{2\pi}}e^{-x^2}$ 的数据表:

X	0.46	0.47	0.48	0.49
$f(X)$	0.322 9	0.319 9	0.316 8	0.313 8

试首先用三次样条构造被积函数 $P(x)$,再分别用 quad 和 quadl 函数计算定积分 $I=\int_{0.46}^{0.49}f(t)\mathrm{d}t$,精度为 10^{-4},并将计算结果与精确值比较.

解 方法 1 首先用三次样条构造被积函数 $P(x)$,再分别用 quad 和 quadl 函数计算定积分 I. 在 MATLAB 工作窗口直接输入下面的程序

```
>> a = 0.46,b = 0.49, x = a:0.01:b;
y = [0.3229  0.3199  0.3168  0.3138],
pp = spline(x,y)
[Q,FCNT] = quad(@ ppval,a,b,[],[],pp)
[Q1,FCNT1] = quadl(@ ppval,a,b,[],[],pp)
```

运行后屏幕显示分别用辛普森和 8 阶牛顿 - 科茨公式计算 I 的结果为

```
pp =
        form: 'pp'
      breaks: [0.4600 0.4700 0.4800 0.4900]
       coefs: [3x4 double]
      pieces: 3
       order: 4
         dim: 1
Q =              FCNT =              Q1 =              FCNT1 =
    0.0096           13               0.0096                 18
```

方法 2 计算 I 的精确解 y 及其近似解 y_1. 输入程序

```
>> syms x
f = exp(-x.^2)./(sqrt(2*pi)); F = int(f, 0.46, 0.49)
y = simple(F), y1 = double(y)
```

运行后屏幕显示结果

```
y =
```

```
        1125899906842624/5644425081792261*pi^(1/2)*(erf(49/
100)-erf(23/50))
    y1 =
        0.0096
```

方法 3 首先用三次样条构造被积函数 $P(x)$,再分别用辛普森和 8 阶牛顿 - 科茨公式计算定积分 I. 在 MATLAB 工作窗口输入程序

```
>>X =[0.46,0.47,0.48,0.49];
Y = [0.3229  0.3199  0.3168  0.3138];
s =interp1 (X,Y, X, 'spline'); L3 =poly2sym (s)
```

运行后输出多项式 $L3$. 保存名为 L3.m 的 M 文件

```
function  L3 =L3(x)
L3 =3229/10000*x.^3 +3199/10000*x.^2 +198/625*x +1569/
5000;
```

然后输入程序

```
[Q1,FCNT1] = quadl(@ L3, 0.46, 0.49,1.e-4)
[Q,FCNT] = quad(@ L3, 0.46, 0.49,1.e-4)
```

运行后屏幕显示计算结果

```
Q1 =          FCNT1 =          Q =              FCNT =
    0.0171           18            0.0171                13
```

由上面输出的结果可以看出,虽然方法 1 和方法 3 都是先用三次样条构造被积函数,再分别用 quad 和 quadl 函数计算定积分 I,但是计算结果不同. 方法 1 的计算结果与精确解 y 的近似解 y_1 相同,但是方法 3 计算结果与 y_1 却不同,为什么呢? 因为方法 1 是用分段三次样条函数逼近被积函数,所以误差小,而方法 3 在积分区间上只用一个三次函数 $L3$ 代替被积函数 $f(x)$,所以误差大,这是我们在计算中应该注意的问题.

9.3.8 利用拉格朗日插值等方法求表格型数值积分的 MATLAB 方法

利用拉格朗日插值多项式求表格型数值积分的 MATLAB 方法与用三次样条的方法有很大的差别. 下面通过例题具体说明前者的解题步骤及二者在解题方法上的差异.

例 9.3.19 测得定积分 $I=\int_0^{\frac{\pi}{2}} f(x)\,dx$ 中被积函数 $Y=f(x)$ 在积分变量 x 的某几个特定点 X =0,0.157 1, 0.471 2, 0.549 8, 0.628 3, 0.863 9, 1.021 0, 1.099 6, 1.570 8 处的值分别为 $Y=$ 0, 0.156 5, 0.454 0,0.522 5, 0.587 8, 0.760 4, 0.852 6, 0.891 0, 1.000 0,试根据这些值首先分别用分段二、三阶拉

格朗日插值多项式和三次样条插值函数构造被积函数,然后分别用 quad 函数和 quad8 函数计算,取精度为 10^{-4},并将计算结果与精确值 1 进行比较.

解 下面分别用分段二阶和三阶拉格朗日插值多项式、三次样条插值方法分别构造被积函数,计算数值积分.

方法 1 (1) 拉格朗日插值多项式的 MATLAB 主程序

```
function L = lfun(X,Y)
m = length(X); n = M1; L = ones(m,m);
for k =1:n +1
     V =1;
     for i =1:n +1
       if k~ =i
     V =conv(V,poly(X(i)))/(X(k) -X(i));
   end
 end
l(k,:) =poly2sym (V);
end
L =Y* l;
```

(2) 用二阶拉格朗日插值多项式构造被积函数. 输入程序

```
>> X =[0,0.1571, 0.4712,0.5498,0.6283,0.8639, 1.0210,1.0996,
1.5708];
Y = [0,0.1565,0.4540,0.5225,0.5878,0.7604,0.8526,0.8910,
1.0000];
L1 =lfun(X(1:3),Y(1:3)), L2 =lfun(X(3:5),Y(3:5)),
L3 =lfun(X(5:7),Y(5:7)), L4 =lfun(X(7:9),Y(7:9)),
```

运行后输出多项式 L_1,L_2,L_3,L_4.

(3) 输入程序

```
>> syms x
L1 _ inline('-14644281526214207/140737488355328000*x.^2 +
2280009553133670687/2251799813685248000*x');
L2 = inline('-88810055001957943/351843720888320000*x.^2 +
15892292661979563/ 14073748835532800 * x - 15511422992738729/
703687441776640000');
L3 = inline('-261101353037957127/703687441776640000*x.^2 +
362054074498830351/ 281474976710656000 * x - 207985426949036897/
281474976710656000');
L4 = inline('-131688789655946847/281474976710656000*x.^2 +
104193434945799191/ 70368744177664000 * x - 930974059276412576875 9/
542958193200437657600 00');
```

```
[Q41,FCNT41] = quad(L1,0, 0.4712,1.e-4);
[Q42,FCNT42] =quad(L2, 0.4712, 0.6283,1.e-4);
[Q43,FCNT43] = quad(L3, 0.6283, 1.0210,1.e-4);
[Q44,FCNT44] = quad(L4, 1.0210, pi/2,1.e-4);
Q = Q41 + Q42 + Q43 + Q44,
FCNT = FCNT41 + FCNT42 + FCNT43 + FCNT44
[Q841,FCNT841] = quad8(L1,0, 0.4712,1.e-4);
[Q842,FCNT842] =quad8(L2, 0.4712, 0.6283,1.e-4);
[Q843,FCNT843] = quad8(L3, 0.6283, 1.0210,1.e-4);
[Q844,FCNT844] = quad8(L4, 1.0210, pi/2,1.e-4);
Q8 = Q841 + Q842 + Q843 + Q844,
FCNT8 = FCNT841 + FCNT842 + FCNT843 + FCNT844
```

运行后屏幕显示分别用 quad 函数和 quad8 函数计算 I 的近似值 Q,Q_8和迭代次数 $FCNT,FCNT8$,取精度为 10^{-4}如下

```
Q =                    FCNT =       Q8 =                   FCNT8 =
0.99957595925413         52         0.99957595925413         132
```

方法 2 (1) 用三阶拉格朗日插值多项式构造被积函数.输入程序

```
>> X =[0,0.1571, 0.4712,0.5498,0.6283,0.8639, 1.0210,1.0996,
1.5708];
Y = [0, 0.1565, 0.4540, 0.5225, 0.5878, 0.7604, 0.8526, 0.8910,
1.0000];
L1 = lfun(X(1:4),Y(1:4)), L2 = lfun(X(4:7),Y(4:7)),
L3 = lfun(X(7:9),Y(7:9)),
```

运行后输出多项式 L_1,L_2,L_3.

(2) 输入程序

```
>> syms x
L1 = inline('-9070382563990323/56294995342131200 * x.^3 -
793956227651389/ 281474976710656000 * x.^2 + 22531519616977366 91/
2251799813685248000 * x');
L2 = inline('-4111817014606 5463/3518437208883 20000 * x.^3 -
1700780706742359/ 21990232555520000 * x.^2 + 7352082157545498 39/
7036874417766400 00 * x -2568820849677 9769/2814749767106560000');
L3 =
  inline ('-13168878965594 6847/ 281474976710656000 * x.^2 +
1041934349457991 91/ 70368744177664000 * x - 9309740592764125768759/
5429581932004376576 0000');
[Q41,FCNT41] = quad(L1,0, 0.5498,1.e-4);
[Q42,FCNT42] =quad(L2, 0.5498, 1.0210,1.e-4);
```

```
[Q43,FCNT43] = quad(L3, 1.0210, pi/2,1.e-4);
Q = Q41 + Q42 + Q43,
FCNT = FCNT41 + FCNT42 + FCNT43
[Q841,FCNT841] = quad8(L1,0, 0.5498,1.e-4,2);
[Q842,FCNT842] -quad8(L2, 0.5498, 1.0210,1.e-4,2);
[Q843,FCNT843] = quad8(L3, 1.0210, pi/2,1.e-4,2);
Q8 - Q841 + Q842 + Q843,
FCNT8 = FCNT841 + FCNT842 + FCNT843
```

运行后屏幕显示分别用 quad 函数和 quad8 函数计算 I 的近似值 Q,Q_{84} 和迭代次数 $FCNT,FCNT8$,取精度为 10^{-4} 如下

```
Q -                         FCNT =
    0.99975250938584          39
Q84 =                       FCNT8 =
    0.99975250938584          99
```

方法 3 三次样条插值多项式计算数值积分. 输入程序

```
>> a =0,b =1.5708,
X =[0,0.1571, 0.4712,0.5498,0.6283,0.8639, 1.0210,1.0996,
1.5708];
Y =[0,0.1565,0.4540,0.5225,0.5878,0.7604,0.8526,0.8910,
1.0000];
pp = spline(X,Y)
[Q8,FCNT8] = quad8(@ppval,a,b,1.e-4,[],pp)
[Q,FCNT] = quad(@ppval,a,b,[],[],pp)
```

运行后屏幕显示

```
Q8 =                       FCNT8 =    Q =                    FCNT =
  1.00012965800727           33      1.00012956660843          17
```

由上面的计算结果可以看出:三种方法计算数值积分的结果与精确值 1 的误差不大.

说明:方法 3 的结果与方法 1 和方法 2 接近. 但是如果三次样条插值多项式和分段埃尔米特插值计算数值积分用类似于方法 1 和方法 2 的方法做,结果就完全不同了. 请看下面的运算.

(1) 三次样条插值多项式计算数值积分用类似于方法 2 的方法做,输入程序

```
>>  X =[0,0.1571, 0.4712,0.5498,0.6283,0.8639, 1.0210,1.0996,
1.5708];
Y =[0,0.1565,0.4540,0.5225,0.5878,0.7604,0.8526,0.8910,
1.0000];
```

```
s1 = interp1 (X(1:4),Y(1:4), X(1:4), ' spline '); L1 = poly2sym
(s1)
s2 = spline (X(4:7),Y(4:7), X(4:7)); L2 =poly2sym (s2)
s3 = spline (X(7:9),Y(7:9), X(7:9)); L3 =poly2sym (s3)
```

运行后输出多项式 L_1,L_2,L_3. 再输入程序

```
> > syms x
L1 - inline('313/2000*x.^2 +227/500*x +209/400');
L2 = inline('209/400*x.^3 +2939/5000*x.^2 +1901/2500*x +
4263/5000');
[Q41,FCNT41] = quad(L1,0, 0.5498,1.e -4);
[Q42,FCNT42] =quad(L2, 0.5498, 1.0210,1.e -4);
[Q43,FCNT43] = quad(L3, 1.0210, pi/2,1.e -4);
Q = Q41 + Q42 + Q43,
FCNT = FCNT41 + FCNT42 + FCNT43
[Q841,FCNT841] = quad8(L1,0, 0.5498,1.e -4,2);
[Q842,FCNT842] =quad8(L2, 0.5498, 1.0210,1.e -4,2);
[Q843,FCNT843] = quad8(L3, 1.0210, pi/2,1.e -4,2);
Q8 = Q841 + Q842 + Q843,
FCNT8 = FCNT841 + FCNT842 + FCNT843
```

运行后屏幕显示分别用 quad 函数和 quad8 函数计算 I 的近似值 Q,Q_8和迭代次数 *FCNT*,*FCNT*8,取精度为 10^{-4}如下

```
Q -                      FCNT4  =
   3.337330631524 25            39
Q8 =                     FCNT8  =
   3.337330631524 25            99
```

(2) 三次样条插值多项式计算数值积分用类似于方法 2 的方法做,输入程序

```
> >  X =[0,0.1571, 0.4712,0.5498,0.6283,0.8639, 1.0210,1.0996,
1.5708];
Y =[0,0.1565,0.4540,0.5225,0.5878,0.7604,0.8526,0.8910,1.0000];
s1 =interp1 (X(1:4),Y(1:4), X(1:4),'pchip'); L1 =poly2sym (s1)
s2 = interp1 (X(4:7),Y(4:7), X(4:7),'pchip'); L2 =poly2sym (s2)
s3 = interp1 (X(7:9),Y(7:9), X(7:9),'pchip'); L3 =poly2sym (s3)
plot(X, Y,'ro')
hold on
plot(X(1:4), L1,'bo')
hold of
```

运行后输出多项式 L_1,L_2,L_3.

(3) 输入程序

```
>> syms x
L1 = inline('313/2000*x.^2+227/500*x+209/400');
L2 = inline('209/400*x.^3+2939/5000*x.^2+1901/2500*x+
4263/5000');
L3 = inline('4263/5000*x.^2+891/1000*x+1');
[Q41,FCNT41] = quad(L1,0, 0.5498,1.e-4);
[Q42,FCNT42] =quad(L2, 0.5498, 1.0210,1.e-4);
[Q43,FCNT43] = quad(L3, 1.0210, pi/2,1.e-4);
Q = Q41 + Q42 + Q43,
FCNT = FCNT41 + FCNT42 + FCNT43
[Q841,FCNT841] = quad8(L1,0, 0.5498,1.e-4,2);
[Q842,FCNT842] =quad8(L2, 0.5498, 1.0210,1.e-4,2);
[Q843,FCNT843] = quad8(L3, 1.0210, pi/2,1.e-4,2);
Q8 = Q841 + Q842 + Q843,
FCNT8 = FCNT841 + FCNT842 + FCNT843
```

运行后屏幕显示分别用 quad 函数和 quad8 函数计算 I 的近似值 Q,Q_8和迭代次数 $FCNT$,$FCNT8$,取精度为 10^{-4}如下

```
Q =                          FCNT =
    3.33733063152425              39
Q8 =                         FCNT8 =
    3.33733063152425              99
```

由此可见,将类似于方法 1 和方法 2 的处理手法处理三次样条插值多项式和分段埃尔米特插值计算数值积分的问题,计算结果已经毫无实际意义,这说明分段埃尔米特插值和样条插值的 MATLAB 函数不能用类似于三阶拉格朗日插值的方法调用.

例 9.3.20 在某次实验中测得一质点在某几个特定时间 t 的速度 $v=v(t)$ 被列入下表,首先根据下表中给定的数据分别用分段二阶、三阶拉格朗日插值多项式、样条插值构造被积函数,然后分别用 quad 函数和 quad8 函数计算在这段时间内质点的位移,并与精确值 $0.2(e^4-e)\approx 10.375\,973\,640\,937\,04$ 比较.

t(s)	1.000 0	1.500 0	2.000 0	2.500 0	3.000 0	3.500 0	4.000 0
$v(t)$(m/s)	0.543 7	0.896 3	1.477 8	2.436 5	4.017 1	6.623 1	10.919 6

解 因为所求在这段时间[1.000 0,4.000 0]内质点的位移为 $s=\int_1^4 v(t)\,dt$,所以下面分别用分段二阶、三阶拉格朗日插值多项式和样条插值方法分别构造被积函数,计算数值积分.

方法 1 （1）用二阶拉格朗日插值多项式构造被积函数. 输入程序

```
>> t =[1.000, 1.5000, 2.0000, 2.5000, 3.0000, 3.5000, 4.0000];
v =[ 0.5437,0.8963,1.4778,2.4365,4.0171,6.6231,10.9196];
L1 =lfun(t(1:3),v(1:3)), L2 =lfun(t(3:5),v(3:5)),
L3 =lfun(t(5:7),v(5:7)),
```

运行后输出多项式 L_1, L_2, L_3.

（2）输入程序

```
>> syms x
L1 =inline('2289/5000*x.^2 -4393/10000*x +1313/2500');
L2 =inline('6219/5000*x.^2 -36797/10000*x +1931/500');
L3 =inline('3381/1000*x.^2 -33529/2000*x +14926/625');
[Q41,FCNT41] = quad(L1,1, 2.0000,1.e-4);
[Q42,FCNT42] = quad(L2, 2.0000, 3.0000,1.e-4);
[Q43,FCNT43] = quad(L3, 3.0000, 4.0000,1.e-4);
Q4 = Q41 + Q42 + Q43,
FCNT4 = FCNT41 + FCNT42 + FCNT43
[Q841,FCNT841] = quad8(L1,1, 2.0000,1.e-14,3);
[Q842,FCNT842] =quad8(L2, 2.0000, 3.0000,1.e-14,3);
[Q843,FCNT843] = quad8(L3, 3.0000, 4.0000,1.e-14,3);
Q8 = Q841 + Q842 + Q843,
FCNT8 = FCNT841 + FCNT842 + FCNT843
```

运行后屏幕显示分别用 quad 函数和 quad8 函数计算 I 的近似值 Q_4, Q_8 和迭代次数 $FCNT4, FCNT8$, 取精度为 10^{-4} 如下

```
Q4 =                    FCNT4 =
    10.37944999999999       39
Q8 =                    FCNT8 =
    10.37944999999999       99
```

方法 2 （1）用三阶拉格朗日插值多项式构造被积函数. 输入程序

```
>> t =[1.000, 1.5000, 2.0000, 2.5000, 3.0000, 3.5000, 4.0000];
v =[ 0.5437,0.8963,1.4778,2.4365,4.0171,6.6231,10.9196];
L1 =lfun(t(1:4),v(1:4)), L2 =lfun(t(4:7),v(4:7)),
```

运行后输出多项式 L_1, L_2.

（2）输入程序

```
>> syms x
L1 =inline('1483/7500*x.^3 -54/125*x.^2 +25379/30000*x -17/250');
L2 =inline('2217/2500*x.^3 -7413/1250*x.^2 +156037/10000*
x  3341/250');
[Q41,FCNT41] =quad(L1,1,2.5000,1.e-4);
```

```
[Q42,FCNT42]=quad(L2,2.5000,4.0000,1.e-4);
Q4=Q41+Q42,FCNT4=FCNT41+FCNT42
[Q841,FCNT841]=quad8(L1,1, 2.5000,1.e-4,4);
[Q842,FCNT842]=quad8(L2,2.5000,4.0000,1.e-4,4);
Q8 = Q841 + Q842,FCNT8 - FCNT841 + FCNT842
```

运行后屏幕显示分别用 quad 函数和 quad8 函数计算 I 的近似值 Q_4,Q_8和迭代次数 *FCNT*4,*FCNT*8,取精度为 10^{-4}如下

```
Q4 =                          FCNT4 =
     10.38360000000000              26
Q84 =                         FCNT8 =
     10.38360000000000              66
```

方法 3 三次样条插值多项式计算数值积分. 输入程序

```
>> a=1,b=4,
t=[1.000, 1.5000, 2.0000, 2.5000, 3.0000, 3.5000, 4.0000];
v=[ 0.5437,0.8963,1.4778,2.4365,4.0171,6.6231,10.9196];
pp= spline (t,v);
[Q8,FCNT8]=quad8(@ ppval,a,b,[],[],pp)
[Q,FCNT]=quad(@ ppval,a,b,[],[],pp)
```

运行后屏幕显示分别用 quad 函数和 quad8 函数计算 I 的近似值 Q,Q_8和迭代次数 *FCNT*,*FCNT*8 如下

```
Q8 =                     FCNT8 =   Q =                    FCNT =
     10.37863802976191      33      10.37862431698066      29
```

由上面的计算结果可以看出:三种方法计算数值积分的结果与精确值 10.375 973 640 937 04很接近.



习 题 9.3

1. 用 MATLAB 的函数 trapz 和 cumtrapz 分别计算 $I=\int_0^{\frac{\pi}{2}} e^{-2x}\sin 5x\,dx$,取 4 位有效数字,估计误差,并与矩形公式比较.

2. 用 MATLAB 函数 trapz 和 cumtrapz 分别计算 $I=\int_0^1 e^{-x^2}dx$,取 4 位有效数字,估计误差,并与精确值比较.

3. 用梯形公式(9.5a)的两种方法分别按列和行计算 $\int_0^n Y(x)dx$,取 4 位有效数字. 其中

(1) $\boldsymbol{Y}=\begin{pmatrix}12 & 21 & 32\\ 21 & 22 & 31\end{pmatrix}$,步长为 1; (2) $\boldsymbol{X}=(1\quad 3\quad 11)$,$\boldsymbol{Y}=(12\quad 22\quad 31)$.

4. 用 rctrap 计算 $I=\frac{1}{\sqrt{2\pi}}\int_0^1 e^{-\frac{x^2}{2}}dx$,递归 17 次,并将计算结果与精确值比较.

5. 用 comsimpson.m 和 quad.m 分别计算 $I=\int_0^{\frac{\pi}{2}} e^{-2x}\sin 5x dx$,取 4 位有效数字,并与梯形公式比较.

6. 用辛普森公式(9.11)和梯形公式(9.5a)分别计算 $I=\int_0^1 e^{-x^2}dx$,取精度分别为 10^{-14} 和 10^{-4},估计误差,并与精确值比较.

7. 用 MATLAB 函数 int 和 quad 计算无界函数的反常积分 $\int_{-1}^1 \frac{1}{\sqrt{x+1}}dx$.

8. 测得定积分 $I=\int_0^{\frac{\pi}{2}} f(x)dx$ 中被积函数 $Y=f(x)$ 在积分变量 x 的某几个特定点 $X=0$, 0.257 2,0.471 2,0.549 8,0.628 3,0.863 9,1.021 0,1.099 6,1.570 8 处的值分别为 $Y=0$, 0.156 5,0.454 0,0.522 5,0.587 8,0.760 4,0.852 6,0.891 0,1.000 0,试根据这些值首先分别用分段二、三阶拉格朗日插值多项式、分段埃尔米特插值、样条插值构造被积函数,然后分别用辛普森和 8 阶牛顿 - 科茨求积公式计算,取精度分别为 10^{-14} 和 10^{-4},并与精确值 1 进行比较.

9. 在某次实验中测得一质点在某几个特定时间 t 的速度 $v=v(t)$ 被列入下表,首先根据下表中给定的数据分别用分段埃尔米特插值、分段二阶、三阶拉格朗日插值多项式、样条插值构造被积函数,然后分别用辛普森和 8 阶牛顿 - 科茨公式计算在这段时间内质点的位移,并与精确值 $0.2(e^4-e)\approx 10.375\,973\,640\,937\,04$ 比较,估计误差.

t(s)	1.00 0	1.500 0	2.000 0	2.500 0	3.000 0	3.500 0	4.000 0
$v(t)$(m/s)	0.443 7	0.796 3	1.877 8	2.936 5	4.017 1	6.623 1	10.919 6

10. 用梯形公式、辛普森和 8 阶牛顿 - 科茨求积公式计算定积分 $I=\int_0^{\frac{\pi}{2}} e^{\sin x}dx$, 取精度 10^{-4},作出它们的积分图,并与精确值进行比较,估计误差.

11. 在某次实验中测得一质点在某几个特定时间 t 的速度 $v=v(t)$ 被列入下表,首先根据下表中给定的数据分别用梯形公式、辛普森、8 阶牛顿 - 科茨公式计算在这段时间内质点的位移,并与精确值 $0.2(e^4-e)\approx 10.375\,973\,640\,937\,04$ 比较,估计误差.

t(s)	1.000 0	1.500 0	2.000 0	2.500 0	3.000 0	3.500 0	4.000 0
$v(t)$(m/s)	0.443 7	0.796 3	1.877 8	2.936 5	4.017 1	6.623 1	10.919 6

12. 用计算 n 阶科茨系数 $C_k^{(n)}$ 和求截断误差公式的 MATLAB 主程序,求计算定积分的近似值的 1 ~ 3 阶牛顿 - 科茨公式的系数和截断误差公式.

13. 用 MATLAB 软件估计用 4、7 阶牛顿 - 科茨公式计算定积分 $I=\int_0^{\frac{\pi}{2}} e^{\sin x}dx$ 的截断误差公式和误差限,取 15 位有小数字.

14. 编写求计算定积分的近似值的 1,2,…,8 阶牛顿 - 科茨公式和系数的 MATLAB 程序

及其误差公式.

9.4 龙贝格(Romberg)公式及其 MATLAB 程序

本节主要介绍递归公式和龙贝格公式及其用软件 MATLAB 计算的方法.

9.4.1 递归公式和龙贝格公式

为了提高复合求积公式的精度,可以利用区间逐次减半的方法.我们看到近似求积公式的误差随着 n 的增加(即步长 h 的减小)而减小,但是对于给定的 $f(x)$ 和精度 ε,在选定了求积公式后如何确定 n 呢?实际上是在求积过程中,用二分法每次将 n 增加一倍,直到满足精度要求.下面以梯形公式为例说明这一过程.

由定理 9.1 和推论 9.1 可知,在 $I=\int_a^b f(x)\mathrm{d}x$ 的积分区间 $[a,b]$ 上插入 $n+1$ 个等距节点 $x_k=a+kh,k=0,1,2,\cdots,n$,取 $h=\dfrac{b-a}{n}$ 时,(9.5a)式产生梯形序列 $\{T_{2^k}\}_{k=0}^{+\infty}$.梯形公式 T_n 的截断误差 $R(f,T_n)$ 为

$$R(f,T_n)=I-T_n=-\frac{b-a}{12}\left(\frac{b-a}{n}\right)^2 f''(\xi_n),\quad \xi_n\in[a,b]$$

当 n 增加一倍,即 h 缩小一半时,梯形公式 T_{2n} 的截断误差 $R(f,T_{2n})$ 为

$$R(f,T_{2n})=I-T_{2n}=-\frac{b-a}{12}\left(\frac{b-a}{2n}\right)^2 f''(\xi_{2n}),\quad \xi_{2n}\in[a,b]$$

即

$$I-T_{2n}\cong\frac{1}{4}(I-T_n),$$

也就是

$$I-T_{2n}\cong\frac{1}{3}(T_{2n}-T_n)\tag{9.37}$$

所以只要 $|T_{2n}-T_n|\leqslant\varepsilon$,计算出的 T_{2n} 即可满足 $|I-T_{2n}|<\varepsilon$ 的精度要求.而每次分点加密一倍时,原分点的函数值 $f(x_k)$ 不需要重新计算,只需求出新分点 (x_k,x_{k+1}) 的中点的函数值 $f_{k+\frac{1}{2}}=f\left[a+(2k+1)\dfrac{b-a}{2n}\right]=f\left[a+\left(k+\dfrac{1}{2}\right)h\right]$,即可算出

$$T_{2n}=\frac{T_n}{2}+\frac{h}{2}\sum_{k=0}^{n-1}f_{k+\frac{1}{2}},\ h=\frac{b-a}{n}\tag{9.38}$$

这样由(9.37)式,得

$$I\cong S_n=\frac{4T_{2n}-T_n}{4-1}\tag{9.39}$$

即由梯形序列$\{T_{2^k}\}_{k=0}^{+\infty}$构造出收敛速度较快的**辛普森序列**$\{S_{2^k}\}_{k=0}^{+\infty}$,(9.39)式也称为**理查森(Richardson)外推公式**.用类似的方法,根据复合辛普森公式的截断误差公式推出

$$C_n=\frac{4^2S_{2n}-S_n}{4^2-1} \tag{9.40}$$

得到收敛速度更快的科茨序列$\{C_{2^k}\}_{k=0}^{+\infty}$

$$C_1,C_2,C_4,C_8,\cdots.$$

在科茨序列$\{C_{2^k}\}_{k=0}^{+\infty}$的基础上还可以重复同样的方法,根据复合科茨公式的截断误差公式推出

$$R_n=\frac{4^3C_{2n}-C_n}{4^3-1} \tag{9.41}$$

得到**龙贝格序列**$\{R_{2^k}\}_{k=0}^{+\infty}$如下

$$R_1,R_2,R_4,R_8,\cdots.$$

为了统一上面的符号,记$n=2^{m-1}$,$T_n=T_{m-1,1}$,则$2n=2^m$,$T_{2n}=T_{m,1}$.将上面的一般形式表示为

$$T_{m,j}=\frac{4^{j-1}T_{m,j-1}-T_{m-1,j-1}}{4^{j-1}-1},j,m=2,3,\cdots\ (m\geqslant j) \tag{9.42}$$

或

$$T_{m,j}=T_{m,j-1}+\frac{T_{m,j-1}-T_{m-1,j-1}}{4^{j-1}-1},j,m=2,3,\cdots\ (m\geqslant j) \tag{9.43}$$

从而得到龙贝格求积公式,将上面的结论概括为下面的定理.

定理 9.4(龙贝格求积公式) 设函数$f(x)$在闭区间$[a,b]$上有定义且在$n+1$个节点$a=x_0<x_1<x_2<\cdots<x_n=b$处的函数值为$f(x_k)$ $(k=0,1,2,\cdots,n)$,则在闭区间$[a,b]$上用二分法每次将n增加一倍,可得到**龙贝格求积公式**

$$\begin{cases} T_{1,1}=\dfrac{h_1}{2}[f(a)+f(b)],h_1=b-a,\\ T_{i,1}=\dfrac{1}{2}T_{i-1,1}+\dfrac{h_{i-1}}{2}\displaystyle\sum_{k=1}^{2^{i-2}}f\left(a+\left(k-\frac{1}{2}\right)h_{i-1}\right),h_i=\dfrac{b-a}{2^{i-1}}=\dfrac{1}{2}h_{i-1},i=2,3,\cdots,\\ T_{m,j}=\dfrac{4^{j-1}T_{m,j-1}-T_{m-1,j-1}}{4^{j-1}-1},\quad j,m=2,3,\cdots\ (m\geqslant j). \end{cases} \tag{9.44}$$

定理 9.5(龙贝格求积公式的误差分析) 设函数$f(x)$在闭区间$[a,b]$上具有$2j$阶连续导数,则在闭区间$[a,b]$上用二分法每次将n增加一倍,得到龙贝格求积公式中的(9.42)式是$2j$**收敛的**,即

$$\int_a^b f(x)\,\mathrm{d}x=T_{m,j}+O(h^{2j}),j,m=2,3,\cdots\ (m\geqslant j), \tag{9.45}$$

截断误差为

$$\int_a^b f(x)\mathrm{d}x - T_{m,j} = O(h^{2j}) = b_j h^{2j} f^{(2j)}(\xi_{m,j}),$$

$$j,m=2,3,\cdots\ (m\geqslant j), \tag{9.46}$$

其中 $h_m=\dfrac{b-a}{2^{m-1}}$，b_j为依赖于 j 的常数，且 $\xi_{m,j}\in[a,b]$.

在实际计算中，并不需要作出序列$\{T_{m,1}\}$后，再作序列$\{T_{m,2}\}$. 而是在算出 $T_{1,1}$和 $T_{2,1}$后就计算 $T_{2,2}$，计算得 $T_{2,2}$和 $T_{3,2}$后就计算 $T_{3,3}$，以此类推. 我们实际上考虑的序列是

$$T_{1,1},T_{2,2},T_{3,3},\cdots,T_{j,j},\cdots.$$

计算 $T_{m,j}$的顺序如下面的龙贝格积分表（见表9-3），可以证明，龙贝格算法是数值稳定的. 当$|T_{j,j+1}-T_{j,j}|\leqslant\varepsilon$ 或$|T_{j+1,j+1}-T_{j,j}|\leqslant\varepsilon$ 时，计算停止. 通过下面的例 9.4.2 和例 9.4.3 可以看到，前者比后者少递归一层，但是后者的精度比前者高.

表 9-3 龙贝格积分表

名称	步长	梯形	辛普森	科茨	龙贝格	第四次改进
误差	h	$O(h^2)$	$O(h^4)$	$O(h^6)$	$O(h^8)$	$O(h^{10})$
1	$b-a$	$T_{1,1}$				
2	$\frac{b-a}{2}$	$T_{2,1}$	$T_{2,2}$			
3	$\frac{b-a}{2^2}$	$T_{3,1}$	$T_{3,2}$	$T_{3,3}$		
4	$\frac{b-a}{2^3}$	$T_{4,1}$	$T_{4,2}$	$T_{4,3}$	$T_{4,4}$	
5	$\frac{b-a}{2^4}$	$T_{5,1}$	$T_{5,2}$	$T_{5,3}$	$T_{5,4}$	$T_{5,5}$

例 9.4.1 用龙贝格求积公式计算 $I=\int_0^{1.5}\dfrac{1}{1+x}\mathrm{d}x$，取精度为 10^{-8}.

解 令 $f(x)=\dfrac{1}{1+x}$，$a=0$，$b=1.5$，根据龙贝格求积公式(9.44)计算得

$$T_{1,1}=\frac{b-a}{2}[f(a)+f(b)]=1.05,$$

$$T_{2,1}=\frac{1}{2}[T_{1,1}+1.5f(0.75)]=0.953\ 571\ 429,$$

$$T_{2,2}=\frac{4T_{2,1}-T_{1,1}}{4-1}=0.921\ 428\ 571,$$

$$T_{3,1}=\frac{1}{2}\{T_{2,1}+0.75[f(0.375)+f(1.125)]\}=0.925\ 983\ 575,$$

$$T_{3,2}=\frac{4T_{3,1}-T_{2,1}}{4-1}=0.916\ 787\ 624,$$

$$T_{3,3}=\frac{4^2T_{3,2}-T_{2,2}}{4^2-1}=0.916\ 478\ 228.$$

以此类推,将计算结果列表 9-4.

表 9-4 例 9.4.1 的龙贝格积分表

i	梯形 $T_{i,1}$	辛普森 $T_{i,2}$	科茨 $T_{i,3}$	龙贝格 $T_{i,4}$	第四次改进 $T_{i,5}$	第五次改进 $T_{i,6}$
1	1.05					
2	0.953 571 429	0.921 428 571				
3	0.925 983 575	0.916 787 624	0.916 478 228			
4	0.918 741 799	0.916 327 874	0.916 297 224	0.916 294 351		
5	0.916 905 342	0.916 293 190	0.916 290 077	0.916 290 776	0.916 290 762	
6	0.916 444 501	0.916 290 888	0.916 290 734	0.916 290 732	0.916 290 732	0.916 290 732

由于 $|T_{6,6}-T_{6,5}|<10^{-8}$,因此

$$I\approx T_{6,6}=0.916\ 290\ 732\approx 0.916\ 290\ 73,$$

从而

$$|I-T_{6,6}|=|\ln 2.5-T_{6,6}|<10^{-8}.$$

9.4.2 龙贝格积分的 MATLAB 程序

用 MATLAB 软件和龙贝格公式计算定积分 $\int_a^b f(x)\,dx$ 的近似值有两种常用的方法:一是根据具体题目编写 MATLAB 程序计算;二是根据(9.44)式编写主程序,然后计算时调用.

下面根据龙贝格求积公式,通过生成 $m>j$ 的龙贝格积分表,并以 $T_{j+1,j+1}$ 作为最终值来逼近 $\int_a^b f(x)\,dx\approx T_{j,j}$. 逼近序列 $T_{m,j}$, $j,m=2,3,\cdots$ ($m\geqslant j$)保存在一个下三角形矩阵中,第 1 列的元

$$T_{1,1}=\frac{h_1}{2}[f(a)+f(b)],\ h_1=b-a,$$

$$T_{i,1}=\frac{1}{2}T_{i-1,1}+\frac{h_{i-1}}{2}\sum_{k=1}^{2^{i-2}}f\left(a+\left(k-\frac{1}{2}\right)h_{i-1}\right),\ i=2,3,\cdots,$$

用于 2^j 个子区间上的连续进行梯形公式的计算,然后利用龙贝格求积公式计算

$T_{m,j}$,其中第 m 行的元为

$$T_{m,j}=\frac{4^{j-1}T_{m,j-1}-T_{m-1,j-1}}{4^{j-1}-1},\ j,m=2,3,\cdots\ (m\geqslant j).$$

当 $|T_{j+1,j+1}-T_{j+1,j}|<wucha$ 时,程序在第 $j+1$ 行停止计算.具体计算数值积分的程序如下:

龙贝格数值积分的 MATLAB 主程序

输入量:*fun* 是被积函数 $f(x)$,a,b 分别是积分下、上限,m 是龙贝格积分表中行的最大数目,*wucha* 是两次相邻迭代值的绝对误差限,即 $|T_{j+1,j+1}-T_{j+1,j}|<wucha$.

输出量:*RT* 是龙贝格积分表,*R* 是利用龙贝格求积公式计算 $\int_a^b f(x)\mathrm{d}x$ 的数值积分值,*wugu* 是误差估计,*h* 是最小步长.

现提供名为 romberg.m 的龙贝格数值积分的 MATLAB 主程序如下:

```
function [RT,R,wugu,h] = romberg(fun,a,b, wucha,m)
n =1;h =b - a; wugu =1; x =a;k =0; RT =zeros(4,4);
RT(1,1) =h *(feval(fun,a) +feval(fun,b))/2;
while((wugu >wucha)&(k <m) |(k <4))
  k =k +1;   h =h/2; s =0;
   for j =1:n
     x =a +h *(2 *j -1); s =s +feval(fun,x);
    end
        RT(k +1,1) = RT(k,1)/2 +h *s; n =2 *n;
    for i =1:k
        RT(k +1,i +1) =((4^i) *RT(k +1,i) -RT(k,i))/(4^i -1);
    end
  wugu =abs(RT(k +1,k) -RT(k +1,k +1));
end
R =RT(k +1,k +1);
```

例 9.4.2 取精度为 $\varepsilon=10^{-8}$,分别用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 和 $|T_{j+1,j+1}-T_{j,j}|<\varepsilon$ 作为计算停止的条件,用 romberg.m 程序计算 $I=\int_0^{1.5}\frac{1}{1+x}\mathrm{d}x$,龙贝格积分表,最小步长 h,取精度为 10^{-8},并与精确值比较.然后取精度为 10^{-7},用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条件,观察用龙贝格求积公式计算的结果与精确值的绝对误差是否满足 10^{-7}.

解 (1) 取精度分别为 10^{-8},当用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条件,输入程序

```
>> F=inline('1./(1+x)');
[RT,R,wugu,h]=romberg(F,0,1.5,1.e-8,13)
syms x
fi=int(1/(1+x),x,0,1.5); Fs=double(fi),
wR=double(abs(fi-R)), wR1 = wR - wugu
```

运行后屏幕显示精确值 I 的近似值 Fs,取精度为 10^{-8},利用龙贝格求积公式的 romberg.m 程序计算 I 的数值解 R,龙贝格积分表 RT,误差估计 $wugu$,最小步长 h,近似值 R 与精确值 f_i 的绝对误差 wR 如下

```
RT =
    Columns 1 through 4
1.05000000000000                 0                 0                 0
0.95357142857143  0.92142857142857                 0                 0
0.92598357524828  0.91678762414057  0.91647822765470                 0
0.91874179909571  0.91632787371152  0.91629722368292 0.9162943506040
0.91690534165717  0.91629318917766  0.91629087687540 0.91629077613242
0.91644450130657  0.91629088785636  0.91629073443494 0.91629073217398
   Columns 5 through 6
         0                 0
         0                 0
         0                 0
         0                 0
 0.91629076211489          0
 0.91629073200160  0.91629073197216
R =                   wugu =                    h =
 0.91629073197216      2.943623123030648e-011    0.04687500000000
Fs =                  wR =                      wR1 =
   0.91629073187416      9.800686628988125e-011     6.857063505957478e-011
```

由此可见,用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条件,递归 6 层,利用龙贝格求积公式的 romberg.m 程序计算 I 的近似值 $R=0.916\ 290\ 73$,所得的相邻两次迭代的绝对误差为 $wugu=2.943\ 623\ 12e-011$,定积分的精确值的近似值 $Fs=0.916\ 290\ 73$,精确值 f_i 与 R 的绝对误差 $wR=9.800\ 686\ 63e-011$,已达到所要求的精度 10^{-8},wR 与 $wugu$ 的绝对误差为 $wR1=6.857\ 063\ 51e-011$,即两者很接近.

如果用 $|T_{j+1,j+1}-T_{j,j}|<\varepsilon$ 作为计算停止的条件,运行的结果如下

```
RT =
    Columns 1 through 4
 1.05000000000000                 0                 0                 0
 0.95357142857143  0.92142857142857                 0                 0
 0.92598357524828  0.91678762414057  0.91647822765470                 0
```

```
0.91874179909571  0.91632787371152  0.91629722368292  0.91629435060401
0.91690534165717  0.91629318917766  0.91629087687540  0.91629077613242
0.91644450130657  0.91629088785636  0.91629073443494  0.91629073217398
0.91632918157305  0.91629074166188  0.91629073191558  0.91629073187559
 Columns 5 through 7
                0                 0                 0
                0                 0                 0
                0                 0                 0
                0                 0                 0
0.91629076211489                  0                 0
0.91629073200160  0.91629073197216                  0
0.91629073187442  0.91629073187429  0.91629073187427
R =                 wugu =                 h =
  0.91629073187427    9.789258292869363e-011    0.02343750000000
wR =                        wR1 =
    1.142833611876187e-013     -9.777829956750600e-011
Fs = 0.916 290 731 874 16
```

递归 7 层,利用龙贝格求积公式的 romberg.m 程序计算 I 的近似值 $R=0.916\ 290\ 73$,所得的相邻两次迭代的绝对误差为 $wugu=9.789\ 258\ 29e-011$,定积分的精确值的近似值 $Fs\ =0.916\ 290\ 73$,精确值 f_i 与 R 的绝对误差 $wR=1.142\ 833\ 61e-013$,wR 与 $wugu$ 的绝对误差为 $wR1=-9.777\ 829\ 96e-011$,即两者很接近.由此可见,用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条件计算的结果比 $|T_{j+1,j+1}-T_{j,j}|<\varepsilon$ 少递归一层,并且其结果与精确值的绝对误差满足 10^{-8}.

(2) 取精度分别为 10^{-7},当用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条件,计算结果如下

```
RT =
 Columns 1 through 4
     1.0500 0000000000           0           0           0
              0
              0
              0
0.91629076211489
 R =                  wugu =                  h =
   0.91629076211489     1.401753557672691e-008     0.09375000000000
 Fs =                   wR =                    wR1 =
   0.91629073187416     3.024073362534249e-008     1.622319804861558e-008
```

由此可见,取精度为 10^{-7},用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条件,用龙贝格求积公式计算的结果与精确值的绝对误差满足 10^{-7}.

例 9.4.3 取精度为 $\varepsilon=10^{-5}$,用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条

件,用 romberg.m 程序计算 $I=\int_{-2}^{2} x^3 \sin x\mathrm{d}x$,龙贝格积分表,最小步长 $h=0.125$,并将计算结果与精确值比较.

解 (1) 首先建立并保存名为 f.m 的 M 函数文件:

```
function y = f(x)
         y = x^3 * sin(x);
```

(2) 然后取 $|T_{j+1,j}-T_{j+1,j+1}|<wucha$ 作为计算停止的条件,输入程序

```
>> [RT,R,wugu,h] = romberg(@ f, -2,2,1.e-5,13)
syms x ,fi = int(x^3 * sin(x),x, -2,2), Fs = double (fi)
wR = double (abs(fi - R)), wR1 = wR - wugu
```

运行后屏幕显示计算结果如下

```
RT =
   Columns 1 through 4
 29.09751765842182                   0                   0                   0
 14.54875882921091    9.69917255280727                   0                   0
  8.95732138422125    7.09350890255803    6.91979799254141                   0
  7.90513446422483    7.55440549089270    7.58513193011501    7.59569278626697
  7.66180088583971    7.58068969304466    7.58244197318813    7.58239927545913
  7.60216666191302    7.58228858727079    7.58239518021920    7.58239443747367
 Columns 5 through 6
                 0                   0
                 0                   0
                 0                   0
                 0                   0
7.58234714404420                     0
7.58239441850117    7.58239446471276
 R =                 wugu =                        h =
   7.58239446471276       4.621159011009013e-008     0.12500000000000
 fi =                       Fs =                   wR =
    8*cos(2)+12*sin(2)     7.58239442953104    3.518172287731917e  008
wR1 =
    -1.102986723277096e-008
```

即 $I=8\cos 2+12\sin 2\cong 7.582\ 39$,取精度为 $\varepsilon=10^{-5}$,用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条件,用 romberg.m 程序计算 I 的近似值 $R=7.582\ 39$,龙贝格积分表为 RT,最小步长 $h=0.125$,R 与精确值 I 的绝对误差 $wR=3.518\ 17\mathrm{e}-008$,满足精度 $\varepsilon=10^{-5}$.



习 题 9.4

1. 分别用手工和计算机计算,用龙贝格求积公式计算定积分 $I=\int_0^{\frac{\pi}{2}} e^{\sin x}\,dx$,取精度 10^{-4},作出它们的积分图,并与精确值进行比较,估计误差.

2. 在某次实验中测得一质点在某几个特定时间 t 的速度 $v=v(t)$ 被列入下表,首先根据下表中给定的数据,用龙贝格求积公式计算定积分在这段时间内质点的位移,并与精确值 $0.2(e^4-e)\approx 10.375\,973\,640\,937\,04$ 比较,估计误差.

t(s)	1.000 0	1.500 0	2.000 0	2.500 0	3.000 0	3.500 0	4.000 0
$v(t)$(m/s)	0.443 7	0.796 3	1.877 8	2.936 5	4.017 1	6.623 1	10.919 6

3. 用龙贝格求积公式计算 $\int_0^{1.5}\frac{1}{1+2x}dx$,取精度为 10^{-7},估计误差.

4. 用龙贝格求积公式计算 $I=\frac{1}{\sqrt{2\pi}}\int_0^1 e^{-\frac{x^2}{2}}dx$,递归 17 次,估计误差,并将计算结果与精确值比较.

5. 分别用复合辛普森公式与复合梯形公式计算积分 $\int_2^4 2\cos x\,dx$,要求准确到 10^{-6},并估计误差.

6. 取精度为 $\varepsilon=10^{-8}$,分别用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 和 $|T_{j+1,j+1}-T_{j,j}|<\varepsilon$ 作为计算停止的条件,用 romberg.m 程序计算 $I=\int_0^1\frac{\sin x}{x}dx$,龙贝格积分表,最小步长 h,取精度为 10^{-8},并与精确值比较.然后取精度为 10^{-7},用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条件,观察用龙贝格求积公式计算的结果与精确值的绝对误差是否满足小于 10^{-7}.

7. 取精度为 $\varepsilon=10^{-5}$,用 $|T_{j+1,j+1}-T_{j+1,j}|<\varepsilon$ 作为计算停止的条件,用 romberg.m 程序计算 $\int_1^2 x^{-1}e^x\,dx$,龙贝格积分表,最小步长 h,并与精确值比较.

9.5 自适应积分及其 MATLAB 程序

为了提高复合积分法的效率,人们还提出了一种自适应算法.因为复合积分法需要使用等距节点,但是,如果被积函数在某些子区间上性态好(如函数值变化不大等),而在另一些子区间上性态不好(如函数值变化巨大等),所以应该在被积函数性态差的子区间上插入较多的节点,在被积函数性态好的子区间上使用较少的节点.自适应积分法的基本思想是通过数值方法辨识被积函数的性态,进而进行合理地剖分.自适应积分法的基础是辛普森求积公式.

假定我们要计算定积分

$$I=\int_a^b f(x)\,\mathrm{d}x$$

的近似值,使误差不超过预先给定的容限 ε. 用自适应积分法数值积分的具体步骤如下:

在第 k 个区间 $[a_k,b_k]$ 上的辛普森求积公式为

$$S(a_k,b_k)=\frac{h}{3}(f(a_k)+4f(c_k)+f(b_k)), \tag{9.47}$$

其中 $c_k=\dfrac{a_k+b_k}{2}$, 且 $h=\dfrac{b_k-a_k}{2}$. 如果被积函数 $f(x)$ 在区间 $[a_k,b_k]$ 上具有四阶连续的导数,则至少存在一个数 $\xi_1\in[a_k,b_k]$, 使得

$$\int_{a_k}^{b_k} f(x)\,\mathrm{d}x=S(a_k,b_k)-\frac{h^5}{90}f^{(4)}(\xi_1). \tag{9.48}$$

将区间 $[a_k,b_k]$ 划分为两个相等的子区间 $[a_{k1},b_{k1}]$ 和 $[a_{k2},b_{k2}]$, 并在每段上递归地使用辛普森求积公式(9.47)实现,每次计算只需要增加计算两个 $f(x)$ 的值,得

$$S(a_{ki},b_{ki})=\frac{h}{6}(f(a_{ki})+4f(c_{ki})+f(b_{ki}))\ \ (i=1,2), \tag{9.49}$$

其中 $a_{k1}=a_k, b_{k1}=a_{k2}=c_k, b_{k2}=b_k, c_{ki}$ 是 $[a_{ki},b_{ki}]$ $(i=1,2)$ 的中点. 则(9.49)式的和为

$$\begin{aligned}&S(a_{k1},b_{k1})+S(a_{k2},b_{k2})\\&=\frac{h}{12}\left[f(a_k)+4f\left(\frac{a_k+c_k}{2}\right)+2f\left(\frac{a_k+b_k}{2}\right)+4f\left(\frac{b_k+c_k}{2}\right)+f(b_k)\right].\end{aligned} \tag{9.50}$$

如果被积函数 $f(x)$ 在区间 $[a_k,b_k]$ 上具有四阶连续的导数,则

$$\int_{a_k}^{b_k} f(x)\,\mathrm{d}x=S(a_{k1},b_{k1})+S(a_{k2},b_{k2})-\frac{1}{4^2}\cdot\frac{h^5}{90}f^{(4)}(\xi_2). \tag{9.51}$$

设 $f^{(4)}(\xi_1)\approx f^{(4)}(\xi_2)$, 则由(9.50)和(9.51)式,得

$$S(a_k,b_k)-\frac{h^5}{90}f^{(4)}(\xi_1)=S(a_{k1},b_{k1})+S(a_{k2},b_{k2})-\frac{1}{4^2}\cdot\frac{h^5}{90}f^{(4)}(\xi_2).$$

移项整理,得

$$-\frac{h^5}{90}f^{(4)}(\xi_2)\approx\frac{4^2}{4^2-1}[S(a_{k1},b_{k1})+S(a_{k2},b_{k2})-S(a_k,b_k)]. \tag{9.52}$$

将(9.52)式代入(9.51)式中,得

$$\begin{aligned}\int_{a_k}^{b_k} f(x)\,\mathrm{d}x\approx\ &S(a_{k1},b_{k1})+S(a_{k2},b_{k2})+\\&\frac{1}{4^2-1}[S(a_{k1},b_{k1})+S(a_{k2},b_{k2})-S(a_k,b_k)].\end{aligned} \tag{9.53}$$

(9.53)式的误差估计公式为

$$\left|\int_{a_k}^{b_k} f(x)\,\mathrm{d}x - S(a_{k1},b_{k1}) - S(a_{k2},b_{k2})\right|$$
$$\approx \frac{1}{4^2-1}\left|S(a_{k1},b_{k1}) + S(a_{k2},b_{k2}) - S(a_k,b_k)\right|. \tag{9.54}$$

自适应积分或者通过辛普森求积公式(9.47)和(9.50)来实现,或者通过辛普森求积公式(9.49),(9.50)和(9.53)来实现.在区间$[a_k,b_k]$上,要使$\int_{a_k}^{b_k} f(x)\,\mathrm{d}x \approx S(a_{k1},b_{k1}) + S(a_{k2},b_{k2})$的误差限为$\varepsilon_k(\varepsilon_k>0)$,由(9.53)式知,只要使

$$\left|S(a_{k1},b_{k1}) + S(a_{k2},b_{k2}) - S(a_k,b_k)\right| < \varepsilon_k \tag{9.55}$$

即可,所以在编程序时,如果$|S(a_{k1},b_{k1}) + S(a_{k2},b_{k2}) - S(a_k,b_k)| < \varepsilon_k$时,则停止运算.

如果需要更准确的精度,可以取$\{[a_0,b_0],\varepsilon_0\}$,其中$\varepsilon_0$为区间$[a_0,b_0]$上的数值积分的误差,将$[a_0,b_0]$细分为两个子区间$[a_{01},b_{01}]$和$[a_{02},b_{02}]$,如果满足(9.55)式,则停止运算.否则,这两个子区间记作$[a_1,b_1]$和$[a_2,b_2]$,取其上的容差为$\varepsilon_1=\varepsilon_2=\frac{\varepsilon_0}{2}$,如此下去,直到满足(9.55)式,则停止运算.

MATLAB 系统提供的 quad.m 程序就是根据自适应积分编写的.在程序 quad.m 中,通过辛普森求积公式(9.49),(9.50)和(9.53)和取$\varepsilon_k = tol$,即用户预先给定的数值积分的精度 *tol*,或默认值10^{-6},如果满足(9.55)式,则停止运算.

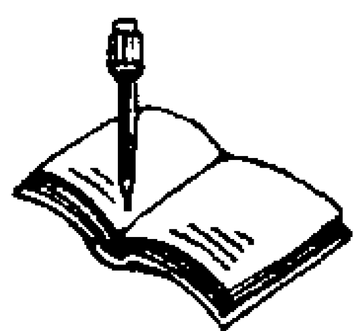
例 9.5.1 用 quad.m 程序计算$\int_0^{1.5}\frac{1}{1+x}\mathrm{d}x$,取精度为$10^{-7}$,并与精确值作比较.

解 输入程序

```
>> F = inline('1./(1+x)'); [Q,FCNT] = quad(F,0,1.5,1.e-7,3)
syms x
fi = int(1/(1+x),x,0,1.5);
Fs = double (fi), wQ = double (abs(fi -Q))
```

运行后屏幕显示 I 精确值 Fs,取精度为10^{-7},用函数 quad 计算 I 的近似值 Q,递归次数 $FCNT$ 它与精确值 Fs 的绝对误差 wQ 如下

```
Q =                             FCNT =
   0.91629073716847                 29
Fs =                            wQ =
    0.91629073187416            5.294317803384376e-009
```



习 题 9.5

1. 用自适应积分计算定积分 $I=\int_0^{\frac{\pi}{2}} e^{\sin x}dx$,取精度 10^{-4},作出它们的积分图,并与精确值进行比较,估计误差.

2. 在某次实验中测得一质点在某几个特定时间 t 的速度 $v=v(t)$ 被列入下表,首先根据下表中给定的数据,用自适应积分计算定积分在这段时间内质点的位移,并与精确值 $0.2(e^4-e)\approx 10.375\ 973\ 640\ 937\ 04$ 比较,估计误差.

t(s)	1.000 0	1.500 0	2.000 0	2.500 0	3.000 0	3.500 0	4.000 0
$v(t)$(m/s)	0.443 7	0.796 3	1.877 8	2.936 5	4.017 1	6.623 1	10.919 6

3. 用自适应积分计算 $\int_0^{1.5}\frac{1}{1+2x}dx$,取精度为 10^{-7}.

4. 用自适应积分计算 $I=\frac{1}{\sqrt{2\pi}}\int_0^1 e^{-\frac{x^2}{2}}dx$,递归 7 次,并将计算结果与精确值比较.

9.6 高斯(Gauss)型积分公式及其 MATLAB 程序

我们首先介绍代数精度的概念,再介绍在[-1,1]上的高斯-勒让德积分公式,然后分别介绍常用的高斯求积公式、拉道(Radau)和洛巴托(Lobatto)积分公式及其 MATLAB 程序.

9.6.1 代数精度

从以上介绍的矩形公式、梯形公式和辛普森公式等数值积分公式我们看到,不论哪个近似求积公式都可以写成如下形式

$$I_n=\sum_{k=1}^{n}A_kf(x_k), \tag{9.56}$$

其中 A_k 是与函数 $f(x)$ 无关的常数,矩形公式、梯形公式和辛普森公式等数值积分公式的区别仅在于 A_k 的取值不同.代数精度是衡量求积公式精确程度的一种指标,它用幂函数作为被积函数 $f(x)$,以近似积分与精确值是否相等作为精度的度量标准,有如下定义:

定义 9.1 设 $f(x)=x^k$,用(9.56)式计算 $I=\int_a^b f(x)dx$,若对于 $k=0,1,\cdots,m$ 都有 $I_n=I$,而当 $k=m+1$ 时 $I_n\neq I$,则称 I_n 的**代数精度为** m.

例 9.6.1 确定系数 A_{-1},A_0,A_1,使求积公式

$$\int_{-h}^{h} f(x)\,dx \approx A_{-1} f(-h) + A_0 f(0) + A_1 f(h) \tag{9.57}$$

具有尽可能高的代数精度,并指出所得求积公式的代数精度.

解 要使求积公式(9.57)至少具有2次代数精度,其充分必要条件为

当$f(x)=1$时,$\int_{-h}^{h} 1\,dx = A_{-1}+A_0+A_1=2h$,

当$f(x)=x$时,$\int_{-h}^{h} x\,dx = A_{-1}(-h)+A_0\cdot 0+A_1\cdot h=0$,

当$f(x)=x^2$时,$\int_{-h}^{h} x^2\,dx = A_{-1}(-h)^2+A_0\cdot 0^2+A_1 f\cdot h^2=\frac{2h^3}{3}$,

即

$$\begin{cases} A_{-1}+A_0+A_1=2h, \\ (-A_{-1}+A_1)h=0, \\ (A_{-1}+A_1)h^2=\frac{2h^3}{3}. \end{cases}$$

解得

$$A_{-1}=A_1=\frac{h}{3},\ A_0=\frac{4h}{3}.$$

代入求积公式(9.57),得

$$\int_{-h}^{h} f(x)\,dx \approx \frac{h}{3}[f(-h)+4f(0)+f(h)]. \tag{9.58}$$

当$f(x)=x^3$时,求积公式(9.58)的左边$=\int_{-h}^{h} f(x)\,dx=\int_{-h}^{h} x^3\,dx=0$,(9.58)式的右边$=\frac{h}{3}[(-h)^3+4\cdot 0^3+h^3]=0$,左边=右边;

当$f(x)=x^4$时,求积公式(9.58)的左边$=\int_{-h}^{h} f(x)\,dx=\int_{-h}^{h} x^4\,dx=\frac{x^5}{5}\Big|_{-h}^{h}=\frac{2h^5}{5}$,(9.58)式的右边$=\frac{h}{3}[(-h)^4+4\cdot 0^4+h^4]=\frac{2h^5}{3}$,左边≠右边;

所以,当求积公式(1)中求积系数取为$A_{-1}=A_1=\frac{h}{3}$, $A_0=\frac{4h}{3}$时,得到求积公式(9.58),其代数精度取到最高,此时代数精度为3.

例9.6.2 求证:梯形公式$T(f)=\frac{b-a}{2}[f(a)+f(b)]$的代数精度$m=1$.

证明 因为 当$f(x)=x^0=1$时,$\int_a^b 1\,dx=b-a=T(1)=\frac{b-a}{2}(1+1)$,

当$f(x)=x$时,$\int_a^b x\,dx=\frac{1}{2}(b^2-a^2)=T(x)=\frac{b-a}{2}(b+a)$,

当 $f(x)=x^2$ 时，$\int_a^b x^2\mathrm{d}x=\frac{1}{3}(b^3-a^3)\neq T(x^2)=\frac{b-a}{2}(b^2+a^2)$，
所以，梯形公式 $T(f)$ 的代数精度 $m=1$.

类似的可以证明，辛普森求积公式 $S(f)=\frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$ 的代数精度 $m=3$.

上面几种求积公式的共同点是将积分区间等分，将分点作为插值节点，用分段插值多项式代替 $f(x)$ 做积分，因而节点数 n 给定后节点 x_k $(k=1,2,\cdots,n)$ 是固定的，要构造求积公式只需确定(9.56)式中的系数 A_k 即可. 这样做虽然简单，但代数精度低. 下面介绍的方法取消区间等分的限制，n 给定后同时确定节点 x_k 和系数 A_k，使代数精度尽量高.

9.6.2　在[−1,1]上的高斯－勒让德(Gauss-Legendre)积分公式及其 MATLAB 程序

先介绍两点高斯－勒让德积分公式，然后再推广到 n 个节点的情形，最后给出 MATLAB 计算程序.

(一) 两点高斯－勒让德积分公式

先看 2 个节点的情况. 因为只有 2 个节点，所以按照固定节点的办法只能用梯形公式，代数精度为 1. 而用下面的方法在区间内适当地选择节点，就可以得到具有 3 阶代数精度的公式. 区间 (a,b) 经过适当的变量代换可以化为 $(-1,1)$，因而只需计算 $I=\int_{-1}^{1}f(x)\mathrm{d}x$.

要构造形如

$$G_2(f)=A_1f(x_1)+A_2f(x_2) \tag{9.59}$$

的求积公式，确定 x_1,x_2,A_1,A_2 使 G_2 的代数精度为 3. 按照代数精度的定义，要求对于 $f(x)=1,x,x^2,x^3$，

$$\int_{-1}^{1}f(x)\mathrm{d}x=A_1f(x_1)+A_2f(x_2)$$

成立，将 $f(x)$ 代入计算可得

$$\begin{cases}A_1+A_2=2,\\ A_1x_1+A_2x_2=0,\\ A_1x_1^2+A_2x_2^2=\dfrac{2}{3},\\ A_1x_1^3+A_2x_2^3=0.\end{cases}$$

解出 $x_1=\frac{-1}{\sqrt{3}},x_2=\frac{1}{\sqrt{3}},A_1=A_2=1$，代入(9.59)即得代数精度为 3 的高斯公式

$$G_2(f)=f\left(-\frac{1}{\sqrt{3}}\right)+f\left(\frac{1}{\sqrt{3}}\right). \tag{9.60}$$

定理 9.6（两点高斯－勒让德积分公式） 如果函数 $f(x)$ 在区间 $[-1,1]$ 上连续，则有代数精度为 $n=3$ 的两点高斯－勒让德积分公式(9.60).

如果函数 $f(x)$ 在区间 $[-1,1]$ 上具有 4 阶连续导数，则

$$\int_{-1}^{1} f(x)\,\mathrm{d}x=G_2(f)+R_2[f],$$

其中两点高斯－勒让德积分公式(9.60)的截断误差为

$$R_2[f]=\frac{f^{(4)}(\xi)}{135}\quad(-1<\xi<1).$$

例 9.6.3 分别用两点高斯－勒让德积分公式(9.60)、步长为 2 的梯形公式和步长为 1 的辛普森求积公式计算 $I=\int_{-1}^{1}\frac{1}{5+x}\mathrm{d}x$，并将计算结果与精确值进行比较.

解 设 $f(x)=\frac{1}{5+x}$，则由两点高斯－勒让德积分公式(9.60)，得

$$G_2(f)=f\left(-\frac{1}{\sqrt{3}}\right)+f\left(\frac{1}{\sqrt{3}}\right)=\frac{1}{5-\frac{1}{\sqrt{3}}}+\frac{1}{5+\frac{1}{\sqrt{3}}}$$

$$=0.405\ 405\ 405\ 405\ 41.$$

由步长为 2 的梯形公式，得

$$T_2=f(-1)+f(1)=0.416\ 666\ 666\ 666\ 67.$$

由步长为 1 的辛普森求积公式，得

$$S_1=\frac{f(-1)+4f(0)+f(1)}{6}=\frac{1}{6}\left(\frac{1}{4}+\frac{4}{5}+\frac{1}{6}\right)$$

$$=0.202\ 777\ 777\ 777\ 78.$$

输入程序

```
>> x1 = -1/sqrt(3);x2 =1/sqrt(3); y1 =1/(5 +x1); y2 =1/(5 +x2);
G2 =y1 +y2, t = -1:2:1;y =1./(5 +t); T =trapz(t,y)
syms x
fi =int(1/(5 +x),x, -1,1); Fs = double (fi), S1 =(1/4 +4/5 +1/6)/6;
wS1 = double (abs(fi -S1)), wG2 = double (abs(fi -G2)),
wT = double (abs(fi -T))
```

运行后屏幕显示如下

```
G2 =                              T =
     0.40540540540541                 0.41666666666667
Fs =                              wS1 =
```

```
    0.40546510810816          0.20268733033038
wG2 =                     wT =
    5.970270275897657e-005    0.01120155855850
```

由此可见,用两点高斯-勒让德积分公式比步长为 2 的梯形公式和步长为 1 的辛普森求积公式计算数值积分的误差都小,而且两点高斯-勒让德积分公式只需计算两项,但是步长为 1 的辛普森求积公式需计算三项.

(二) n 个节点的高斯-勒让德积分公式及其误差分析

一般的,n 个节点的高斯-勒让德积分公式的代数精度为 $2n-1$,其数值积分公式为

$$G_n(f)=A_{n,1}f(x_1)+A_{n,2}f(x_2)+\cdots+A_{n,n}f(x_n),\tag{9.61}$$

$$\int_{-1}^{1}f(x)\,\mathrm{d}x=G_n(f)+R_n[f],$$

其中截断误差为

$$R_n[f]=\frac{f^{(2n)}(\xi)}{(2n)!}\int_{-1}^{1}\omega_n^2(x)\,\mathrm{d}x\ (-1<\xi<1),\tag{9.62}$$

其中 $\omega_n(x)=(x-x_1)(x-x_2)\cdots(x-x_n)$.

将所需要的常用的高斯-勒让德积分公式(9.61)中的节点的横坐标 x_k 和系数 $A_{n,k}(k=1,2,\cdots,n)$ 及截断误差 $R_n[f]$ 已经制成表 9-5,用时直接查找即可.

表 9-5 常用的高斯-勒让德积分公式表

$\int_{-1}^{1}f(x)\,\mathrm{d}x=\sum_{k=1}^{n}A_{n,k}f(x_k)+R_n[f]$			
n	横坐标 x_k	系数 $A_{n,k}$	截断误差 $R_n[f]$ $(-1<\xi<1)$
2	$-\frac{1}{\sqrt{3}},\frac{1}{\sqrt{3}}$	1,1	$\frac{f^{(4)}(\xi)}{135}$
3	$-\sqrt{\frac{3}{5}},0,\sqrt{\frac{3}{5}}$	$\frac{5}{9},\frac{8}{9},\frac{5}{9}$	$\frac{f^{(6)}(\xi)}{15\,750}$
4	±0.861 136 311 6 ±0.339 981 043 6	0.347 854 845 1 0.652 145 154 9	$\frac{f^{(8)}(\xi)}{3\,472\,875}$
5	±0.906 179 845 9 ±0.538 469 310 1 0.000 000 000 0	0.236 926 885 1 0.478 628 670 5 0.568 888 888 8	$\frac{f^{(10)}(\xi)}{1\,237\,732\,650}$
6	±0.932 469 514 2 ±0.661 209 386 5 ±0.238 619 186 1	0.171 324 492 4 0.360 761 573 0 0.467 913 934 6	$\frac{2^{13}(6!)^4f^{(12)}(\xi)}{(12!)^3 13!}$

续表

$\int_{-1}^{1} f(x)dx = \sum_{k=1}^{n} A_{n,k} f(x_k) + R_n[f]$			
7	±0.949 107 912 3 ±0.741 531 185 6 ±0.405 845 151 4 0.000 000 000 0	0.129 484 966 2 0.279 705 391 5 0.381 830 050 5 0.417 959 183 7	$\frac{2^{15}(7!)^4 f^{(14)}(\xi)}{(14!)^3 15!}$
8	±0.960 289 856 5 ±0.796 666 477 4 ±0.525 532 409 9 ±0.183 434 642 5	0.101 228 536 3 0.222 381 034 5 0.313 706 645 9 0.362 683 783 4	$\frac{2^{17}(8!)^4 f^{(16)}(\xi)}{(16!)^3 17!}$

（三）n 个节点的高斯－勒让德积分公式及其误差分析的两个 MATLAB 程序

根据式(9.61)和(9.62)编写的用 n 个节点的高斯－勒让德积分公式计算 $\int_{-1}^{1} f(x)dx$ 的数值积分(代数精度为 $2n-1$)及其截断误差公式的 MATLAB 程序如下：

用高斯－勒让德积分公式(9.61)计算 $I=\int_{-1}^{1} f(x)dx$ 的数值积分及其截断误差公式的 MATLAB 主程序

输入量：*fun* 是被积函数 $f(x)$，X 是 n 个节点的横坐标 x_k 的 $1\times n$ 维向量(可以从表 9－5 中取得)，A 是来自表 9－5 中的系数 $A_{n,k}(k=1,2,\cdots,n)$ 的 $1\times n$ 维向量.

输出量：*GL* 是利用 n 个节点的高斯－勒让德积分公式(9.61)计算 I 的数值积分值，代数精度为 $2n-1$，*RGn* 是截断误差公式，其中 M 是 $f(x)$ 的 $2n$ 阶导数，Y 是 $f(x)$ 在 X 处的值.

```
function [GL,Y, RGn] = GaussR1(fun,X,A)
n = length(X);n2 = 2 * n;Y = feval(fun,X);GL = sum(A.* Y);
sun = 1; su2n = 1; su2n1 = 1; wome = 1;
syms x
for k = 1:n
      wome = wome * (x - X(k));
end
wome2 = wome^2;Fr = int(wome2,x, -1,1);
for k = 1:n2
    su2n = su2n * k;
end
```

```
syms M
RGn = Fr * M/su2n;
```

例 9.6.4 用高斯 - 勒让德积分公式(9.61)计算 $I=\frac{1}{\sqrt{2\pi}}\int_{-1}^{1}e^{-\frac{x^2}{2}}dx$，取代数精度为 3 和 5，再根据截断误差公式(9.62)写出误差公式，并将计算结果与精确值进行比较.

解 (1) 建立并保存 fun.m 文件命名的 M 文件函数

```
function y = fun(x)
         y = exp( ( -x.^2)./2)./(sqrt(2 * pi));
```

(2) 建立并保存 GaussR1.m 文件命名的 M 文件函数

(3) 根据表 9-5，输入程序

```
>> X1 =[ -1/(3^(1/2)),1/(3^(1/2))]; A1 =[1,1];
[GL1,Y1, Rn1] =GaussR1(@ fun,X1,A1)
X2 =[ -(3/5)^(1/2),0,(3/5)^(1/2)]; A2 =[5/9,8/9,5/9];
[GL2,Y2, Rn2] =GaussR1(@ fun,X2,A2)
syms x
fi =int(exp( ( -x.^2)./2)./(sqrt(2 * pi)),x, -1,1);
Fs = double (fi), wGL1 = double (abs( fi - GL1)), wGL2 = double
(abs( fi -GL2))
```

运行后屏幕显示分别用代数精度为 3 和 5 的高斯 - 勒让德积分公式数值计算 I 的结果 GL_1 和 GL_2、截断误差公式 Rn_1 和 Rn_2、精确值 Fs_1 和 Fs_2 及其两者的绝对误差 wGL_1 和 wGL_2 依次如下

```
Rn1 =            GL1 =              Y1 =
 1/135 * M           1754/2597          877/2597     877/2597
Rn2 =            GL2 =              Y2 =
   1/15750 * M     237/347           451/1526  1056/2647  451/1526
Fs =            wGL1 =              wGL2 =
  2315/3391         107/14668          36/116971
```

用代数精度为 3 和 5 的高斯 - 勒让德积分公式数值计算 I 的结果 $GL_1=\frac{1\,754}{2\,597}$ 和 $GL_2=\frac{237}{347}$，与精确值 $I\approx Fs=\frac{2\,315}{3\,391}$ 的绝对误差分别为 $wGL_1=\frac{107}{14\,668}$ 和 $wGL_2=\frac{36}{116\,971}$. 根据截断误差公式(9.62)计算的截断误差的公式为

$$R_2[f]=\frac{1}{135}f^{(4)}(\xi),\ R_3[f]=\frac{1}{15\,750}f^{(6)}(\xi)\ (-1<\xi<1).$$

另外，根据(9.61)和(9.62)编写的用 n 个节点的高斯 - 勒让德积分公式计算 $I=\int_{-1}^{1}f(x)dx$ 的数值积分(代数精度为 $2n-1$)及其估计绝对误差限的

MATLAB程序如下：

> **用高斯－勒让德积分公式计算 $\int_{-1}^{1} f(x)\,dx$ 的数值积分和误差估计的 MATLAB 主程序**
>
> 输入量：*fun2n* 是被积函数 $f(x)$ 的 $2n$ 阶导数，*fun* 是被积函数 $f(x)$，$\boldsymbol{X}$ 是 n 个节点的横坐标 x_k 的 $1\times n$ 维向量（可以从表 9－5 中取得），$\boldsymbol{A}$ 是来自表9－5中的系数 $A_{n,k}(k=1,2,\cdots,n)$ 的 $1\times n$ 维向量.
>
> 输出量：*Rn* 是高斯－勒让德积分公式(9.61)的误差估计，*GL* 是公式(9.61)计算 $\int_{-1}^{1} f(x)\,dx$ 的数值积分值，代数精度为 $2n-1$，$\boldsymbol{Y}$ 是被积函数 $f(x)$ 在 $\boldsymbol{X}$ 处的值.

```
function [GL,Y,Rn] = GaussR2 (fun,X,A,fun2n)
n = length(X);n2 = 2 * n; Y = feval(fun,X);
GL = sum(A. * Y); GL = sum(A. * Y);
sun = 1; su2n = 1; su2n1 = 1; wome = 1;
syms x
for k = 1:n
     wome = wome * (x - X(k));
end
wome2 = wome^2;Fr = int(wome2,x, -1,1);
for k = 1:n2
  su2n = su2n * k;
end
 mfun2n = max(fun2n); Rn = Fr * mfun2n/su2n;
```

例 9.6.5 用高斯－勒让德积分公式(9.61)计算 $\int_{-1}^{1} \frac{1}{5+x}dx$，取代数精度为 15，再用截断误差公式(9.62)估计误差，并将计算结果与精确值进行比较.

解 (1) 建立并保存 fun.m 文件命名的 M 文件函数

```
function y = fun(x)
        y = 1./(5 + x);
```

(2) 建立并保存 GaussR2.m 文件命名的 M 文件函数.

(3) 输入程序

```
>> syms x
fun2n = diff(1./(5 + x),x,16)
```

(4) 运行后，再输入程序

```
>> X = [ -0.9602898565, -0.7966664774, -0.5255324099,
-0.1834346425, 0.9602898565, 0.7966664774, 0.5255324099,0.1834346425];
```

```
A = [0.1012285363, 0.2223810345, 0.3137066459, 0.3626837834,
0.1012285363, 0.2223810345, 0.3137066459, 0.3626837834];
x = -1:0.00001:1; fun2n =20922789888000./(5 +x).^17;
[GL,Y,Rn] =GaussR2 (@ fun,X,A,fun2n)
syms t
fi =int(1/(5 +t),t, -1,1);Fs = double (fi),wGL = double(abs(fi -GL))
```

运行后屏幕显示用代数精度为 15 的高斯 - 勒让德积分公式数值计算 I 的结果 GL 和精确值 Fs 及其两者的绝对误差 wGL 依次如下

```
GL =                                  Rn =
   0.40546510814876                        2.709468201849595e-015
Y =
  Columns 1 through 4
0.24754251282336  0.23790641276104  0.22349027674545  0.20761682356139
  Columns 5 through 8
0.16777707528929  0.17251294410310  0.18097803538503  0.19292227431611
Fs =                            wGL =
   0.40546510810816                  4.059186608518054e-011
```

用代数精度为 15 的高斯 - 勒让德积分公式数值计算 I 的结果 $GL=0.405\ 465\ 108\ 1$与精确值 $I\approx Fs=0.405\ 465\ 108\ 1$ 的绝对误差为 $wGL=4.059\ 186\ 608\ 5e-011$. 根据截断误差公式(9.62)估计的绝对误差限为 $R_n=2.709\ 468\ 201\ 8e-015$.

9.6.3 在$[a,b]$上的高斯 - 勒让德积分公式及其 MATLAB 程序

(一) 权 $A_{n,k}=1$ $(k=1,2,\cdots,n)$的高斯 - 勒让德积分公式

进一步提高精度有两条途径,一是增大 n,可以证明,$G_n(f)=\sum_{k=1}^{n}A_kf(x_k)$ 的代数精度可达 $2n-1$, 但是要解由(9.61)式得到的更复杂的非线性方程组,n 较大时实用价值不大;二是先将区间(a,b)分小,在小区间上用 $G_2(f)$,具体做法如下:

将区间$(a,b)m$ 等分,记 $h=\frac{b-a}{m}$,$x_k=a+kh,k=0,1,\cdots,m$,作变换$x=\frac{x_k+x_{k-1}}{2}+\frac{h}{2}t$,则

$$I_k=\int_{x_{k-1}}^{x_k}f(x)\mathrm{d}x=\frac{h}{2}\int_{-1}^{1}f\left(\frac{x_k+x_{k-1}}{2}+\frac{h}{2}t\right)\mathrm{d}t.$$

用 $G_2(f)$近似 I_k,就有

$$I_k \cong \frac{h}{2}[f(z_k^{(1)})+f(z_k^{(2)})],$$

$$z_k^{(1)}=\frac{x_k+x_{k-1}}{2}+\frac{h}{2\sqrt{3}},\ z_k^{(2)}=\frac{x_k+x_{k-1}}{2}-\frac{h}{2\sqrt{3}}.$$

于是

$$\int_a^b f(x)\,\mathrm{d}x \cong \frac{h}{2}\sum_{k=1}^{m}[f(z_k^{(1)})+f(z_k^{(2)})], \tag{9.63}$$

这就是常用的高斯求积公式.

(二) 一般型高斯－勒让德积分公式

用同样的方法可得到下面的定理.

定理 9.7(一般型高斯－勒让德积分公式) 如果函数 $f(x)$ 在区间 $[-1,1]$ 上有 n 个节点,且代数精度为 $2n-1$ 的高斯－勒让德积分公式的节点的横坐标 x_k 和权 $A_{n,k}(k=1,2,\cdots,n)$ 已知,作变换 $t=\frac{a+b}{2}+\frac{b-a}{2}x$,则一般的高斯－勒让德积分公式为

$$\int_a^b f(t)\,\mathrm{d}t \cong \frac{b-a}{2}\sum_{k=1}^{n}A_{n,k}f\left(\frac{a+b}{2}+\frac{b-a}{2}x_k\right). \tag{9.64a}$$

其中截断误差为

$$R_n[f]=\frac{f^{(2n)}(\xi)}{(2n)!}\int_a^b \omega_n^2(x)\,\mathrm{d}x\ (a<\xi<b), \tag{9.64b}$$

其中 $\omega_n(x)=(x-x_1)(x-x_2)\cdots(x-x_n)$.

根据(9.64)编写的用一般型高斯－勒让德积分公式计算 $\int_a^b f(x)\,\mathrm{d}x$ 的数值积分的 MATLAB 程序如下:

用一般型高斯－勒让德积分公式计算 $\int_a^b f(x)\,\mathrm{d}x$ 的数值积分 MATLAB 主程序

输入量:*fun2n* 是被积函数 $f(x)$ 的 $2n$ 阶导数,*fun* 是被积函数 $f(x)$,a,b 分别是积分下、上限,$\boldsymbol{X}$ 是 n 个节点的横坐标 x_k 的 $1\times n$ 维向量(可以从表9－5中取得),$\boldsymbol{A}$ 是来自表 9－5 中的系数 $A_{n,k}(k=1,2,\cdots,n)$ 的 $1\times n$ 维向量.

输出量:*GL* 是利用一般型高斯－勒让德积分公式(9.64)计算 $\int_a^b f(x)\,\mathrm{d}x$ 的数值积分值,$\boldsymbol{Y}$ 是 $f\left(\frac{a+b}{2}+\frac{b-a}{2}x_k\right)$ 的值,*Rn* 是高斯－勒让德积分公式(9.64b)的误差估计.

```
function [GL,Y,Rn] = Gauss (fun,a,b,X,A,fun2n)
```

```
n = length(X); n2 = n * 2; T = zeros(1,n);
T = ((a + b)/2) + ((b - a)/2) * X;
Y = feval(fun,T); GL = ((b - a)/2) * sum(A.* Y);
sun = 1; su2n = 1; su2n1 = 1; wome = 1;
syms x
for k = 1:n
wome - wome * (x  X(k));
end
wome2 = wome^2;Fr = int(wome2,x, -1,1);
for k = 1:n2
su2n = su2n * k;
end
mfun2n = max(fun2n); Rn = Fr * mfun2n/su2n;
```

例 9.6.6 利用一般型高斯 - 勒让德积分公式计算 $I=\frac{1}{\sqrt{2\pi}}\int_0^2 e^{-\frac{x^2}{2}}dx$,取代数精度为 15,并与精确值比较.

解 (1) 建立 fun.m 文件命名的 M 文件函数

```
function y = fun(x)
         y = exp( ( -x.^2)./2)./(sqrt(2 * pi));
```

(2) 输入程序

```
>>X = [ -0.9602898565, -0.7966664774, -0.5255324099,
-0.1834346425, 0.9602898565, 0.7966664774, 0.5255324099,0.1834346425];
A = [0.1012285363, 0.2223810345, 0.3137066459, 0.3626837834,
0.1012285363, 0.2223810345, 0.3137066459, 0.3626837834];
a = 0;b = 2; [GL,Y,Rn] = Gauss (@ fun,a,b,X,A,fun2n)
syms x, fi = int(exp(( -x^2)/2)/(sqrt(2 * pi)),x,a,b);
Fs = double (fi), wGL = double (abs(fi - GL))
```

运行后屏幕显示分别用代数精度为 15 的一般型高斯 - 勒让德积分公式数值计算 I 的结果 GL、高斯 - 勒让德积分公式(9.64b)的误差 Rn、估计被积函数值 Y 和精确值 Fs 及其两者的绝对误差 wGL 依次如下

```
GL =                              Rn =
   0.47724986810026                    2.709468201849595e-015
Y =
 0.39862785922733  0.39077989968078  0.35647246460968  0.28583858238797
0.05840774991356  0.07942486895803  0.12461041358066  0.19805761421327
Fs =                         wGL =
   0.47724986805182          4.843847992335364e-011
```

9.6.4 拉道(Radau)积分公式和洛巴托(Lobatto)积分公式及其 MATLAB 程序

如果预先给定 $m+2n$ 个节点(x_k,y_k)中的 $y_k(k=1,2,\cdots,m+2n)$,要构造形如

$$\int_a^b f(x)\mathrm{d}x = \sum_{k=1}^{m} B_k f(y_k) + \sum_{k=1}^{n} A_k f(x_k) + R[f] \tag{9.65}$$

的求积公式,确定系数 A_k,B_k和 $x_k(k=1,2,\cdots,n)$,使

$$G_n = \sum_{k=1}^{m} B_k f(y_k) + \sum_{k=1}^{n} A_k f(x_k) \tag{9.66}$$

的代数精度为 $m+2n-1$. 按照代数精度的定义,要求对于 $f(x)=1,x,x^2,\cdots,x^{m+2n-1}$,

$$\int_{-1}^{1} f(x)\mathrm{d}x = \sum_{k=1}^{m} B_k f(y_k) + \sum_{k=1}^{n} A_k f(x_k)$$

成立.

有时为了确定系数 A_k,B_k和 $x_k(k=1,2,\cdots,n)$,用著名的勒让德正交多项式

$$P_0=1,P_1=x,\quad P_2=\frac{3x^2-1}{2},$$

$$P_3=\frac{5x^3-3x}{2},\quad P_4=\frac{35x^4-30x+3}{8},$$

$$\cdots\cdots\cdots\cdots$$

$$P_n=\frac{1}{2^n}\sum_{m=0}^{[\frac{n}{2}]}(-1)^m C_n^m C_n^{2(n-m)} x^{n-2m}.$$

下面介绍两种常用的高斯型积分公式,它们都是将区间的端点取作预先给定的节点,分别称作拉道积分公式和洛巴托积分公式.

(一) 拉道积分公式

$$\int_{-1}^{1} f(x)\mathrm{d}x = \frac{2}{n^2}f(-1) + \sum_{k=1}^{n-1} A_k f(x_k) + R[f] \tag{9.67}$$

称为**拉道积分公式**. 其中 x_k是多项式$\frac{P_{n-1}(x)+P_n(x)}{x+1}$的根,$A_k=\frac{1-x_k}{n^2P_{n-1}^2(x_k)}$,$x_k\neq 1$.

误差余项为

$$R[f] = -\frac{2^{n-1}n[(n-1)!]^4}{[(2n-1)!]^3}f^{(2n-1)}(\xi),\quad -1<\xi<1. \tag{9.68}$$

(二) 洛巴托积分公式

$$\int_{-1}^{1} f(x)\,\mathrm{d}x = \frac{2}{n(n-1)}[f(-1)+f(1)] + \sum_{k=2}^{n-1} A_k f(x_k) + R[f] \qquad (9.69)$$

称为**洛巴托积分公式**. 其中 x_k 是多项式 $P'_{n-1}(x)$ 的第 $j-1$ 个根, $A_k = \dfrac{2}{n(n-1)P_{n-1}^2(x_k)}$, $x_k \neq \pm 1$. 误差余项为

$$R[f] = -\frac{2^{n-1}n(n-1)^3[(n-2)!]^4}{(2n-1)[(2n-2)!]^3} f^{(2n-2)}(\xi), \quad -1<\xi<1. \qquad (9.70)$$

(三) 用洛巴托数值积分公式的 MATLAB 程序

对于高斯型积分公式,在实际应用中也常用复合法则和自适应法则. 在 MATLAB 系统中,根据自适应洛巴托求积公式编写了数值积分的程序 quadl.m,是 MATLAB 6.5 和 MATLAB 7.01 以后的版本极力推荐的程序,其功能和调用格式如下:

调用格式一:Q = quadl (FUN,a,b)

此命令用于高阶自适应求函数 $Y = FUN(X)$ 从 a 到 b 的积分,误差为 10^{-6},输入被积函数 FUN 和积分下、上限 a 和 b,运行后输出积分的近似值向量 $\boldsymbol{Q}$. 运行时,被积函数将接受一个以向量形式的自变量 $\boldsymbol{X}$,并且返回在 $\boldsymbol{X}$ 的每个元处的积分的近似值所构成的向量 $\boldsymbol{Q}$.

调用格式二:Q = quadl (FUN,a,b,TOL)

此命令用绝对误差限 TOL 代替默认值 10^{-6}. 其他同上.

调用格式三:[Q,FCNT] = quadl (...)

其中输出量 $FCNT$ 是递归计算的次数. 其他同上.

调用格式四:quadl (FUN,a,b,TOL,TRACE)

输入非零数 $TRACE$,则屏幕显示在递归计算期间的向量[fcnt a b-a Q].其他同上.

调用格式五:quadl (FUN,A,B,TOL,TRACE,P1,P2,...)

其中输入量 $P_1, P_2, \cdots$ 是附加的参数. 其他同上.

调用格式六:quadl FUN(X,P1,P2,...)

对于 TOL 或 $TRACE$ 使用默认值. 其他同上.

说明:(1) 上面所有的函数 FUN 的调用格式,都必须指定被积函数 FUN 为嵌入对象 F = inline('FUN') ,然后调用 Q = quadl(F,a,b)或指定为函数处理,例如,将被积函数 FUN 作一个名为 fun.m 的 M 文件:

```
function y = fun(X)
    y = FUN (X)
```

然后调用 Q = quadl(@ myfun,a,b,...).

(2) 在定义被积函数 *FUN* 中的运算用数组算子. * , ./和.^;

(3) 如果没有给定被积函数 $Y = FUN(X)$ 的具体表达式,而只给定了积分变量和被积函数的离散的数据 $\boldsymbol{X} = (x_0, x_1, \cdots, x_n)$, $\boldsymbol{Y} = (y_0, y_1, \cdots, y_n)$,或数表

x	x_0	x_1	x_2	x_3	…	x_n
$y=f(x)$	y_0	y_1	y_2	y_3	…	y_n

则常用的处理方法如下:

首先利用插值方法(如分段样条插值,分段低阶的拉格朗日插值等)求出分段插值函数 fun,然后用说明(1)中的方法重新指定插值函数 fun.

(4) 在 MATLAB 6.5 中可以计算无界函数的反常积分.

例 9.6.7 用函数 quadl 和 quad8 及 quad 分别数值计算 $I = \dfrac{1}{\sqrt{2\pi}}\displaystyle\int_0^{\frac{\pi}{2}} e^{-\frac{x^2}{2}} dx$,分别取精度为 10^{-6} 和 10^{-14},将计算结果分别与精确值比较,并画出图形.

解 (1) 建立 fun.m 文件命名的 M 文件函数

```
function y = fun(x)
        y = exp( ( -x.^2)./2)./(sqrt(2*pi));
```

(2) 输入程序

```
>>subplot(2,1,1),
[Q1,FCNT1] =quadl(@ fun,0,pi/2,1.e-6)
[Q8,FCNT8] =quad8(@ fun,0,pi/2,1.e-6,3)
[Q,FCNT] =quad(@ fun,0,pi/2,1.e-6)
syms x
fi =int(exp(( -x.^2)./2)./(sqrt(2*pi)),x,0, pi/2);
Fs =double(fi), wQ1 =double(abs(fi -Q1))
wQ8 =double(abs(fi -Q8)), wQ =double(abs(fi -Q))
xlabel('自变量 X'), ylabel('因变量 Y')
title('取精度为1.e-6,被积函数 f(x) =exp( -x^2/2)/(sqrt(2*pi))
在[0,pi/2]上的数值积分')
legend('取精度 1.e-6,用 8 阶牛顿-科茨公式计算积分过程的递归图')
hold on
subplot(2,1,2)
[Q114,FCNT114] =quadl(@ fun,0,pi/2,1.e-14)
[Q814,FCNT814] =quad8(@ fun,0,pi/2,1.e-14,3)
[Q14,FCNT14] =quad(@ fun,0,pi/2,1.e-14,[],1)
wQ114 =double(abs(fi -Q114)), wQ814 =double(abs(fi -Q814))
```

```
wQ14 =double(abs(fi -Q14)),
xlabel('自变量 X'), ylabel('因变量 Y')
title('取精度为 1.e -14,被积函数 f(x) = exp( -x^2/2)/(sqrt(2
pi))在[0,pi/2]上的数值积分')
legend('取精度 1.e -14,用 8 阶牛顿 -科茨公式计算积分过程的递归图')
hold off
```

运行后屏幕显示图 9 -4 和 I 的精确值 F_s,分别取精度为 10^{-6}和 10^{-14},分别用函数 quadl 和 quad8 及 quad 计算 I 的近似值 Q_1,Q_8和 Q_{14},Q_{l14},Q_{814}和 Q_{14},递归次数 $FCNT1$,$FCNT8$ 和 $FCNT$,$FCNTl14$、$FCNT814$ 和 $FCNT14$,近似值分别与精确值f_t的绝对误差 wQ_1,wQ_8和 wQ,wQ_{l14},wQ_{814}和 wQ_{14}如下

```
Q1 =                    FCNT1 =      Q8 =                    FCNT8 =
  0.44188501970377        18          0.44188501721659         33
Q =                     FCNT =        Fs =
  0.44188511819589        13          0.44188501721659
wQ1 =                                wQ8 =
   2.487183090649321e -009            3.572753311048285e -015
wQ =                                 Ql14 =
   1.009793004969618e -007            0.44188501721659,
FCNTl14 =        Q814 =                         FCNT814 =
          138            0.44188501721659              129
Q14 =                     FCNT 14 =        wQl14 =
   0.44188501721659           553           1.310619347103004e -016
wQ814 =                               wQ14 =
   9.098267021473094e -017              2.003963224778471e -017
```

由此可见,取精度为 10^{-6},用 8 阶牛顿 -科茨求积公式,递归 33 次,计算数值积分 0.441 885 017 216 59 与精确值 F_s =0.441 885 017 216 59 的误差 wQ_8 = 3.572 753 311 048 285e -015 最小;用洛巴托求积公式,递归 18 次,计算数值积分 0.441 885 019 703 77 的误差 wQ_1 =2.487 183 090 649 32e -009 与前者相差很大;而用辛普森求积公式,递归 13 次,计算数值积分 Q =0.441 885 118 195 89 的误差 wQ =1.009 793 004 969 62e -007 最大,但是这三种计算结果都达到了预先给定的精度 10^{-6}. 取精度为 10^{-14},用 8 阶牛顿 -科茨求积公式,递归 129 次,计算数值积分 0.441 885 017 216 59 与精确值 Fs 的误差 wQ_{814} = 9.098 267 021 473 094e -017第二小;用洛巴托求积公式,递归 138 次,计算数值积分的误差 wQ_{l14} =1.310 619 347 103 00e -016 最大;而用辛普森求积公式,递归 553 次,计算数值积分 Q_{14}的误差 wQ_{14} =2.003 963 224 778 47e -017 最小,但是这三种计算结果都达到了预先给定的精度 10^{-14}.

说明:图 9 -4 中的点个数等于用函数 quad8 计算数值积分的递归次数,图

中的点是用 quad8 计算 I 的过程中被积函数 $f(x)=\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$ 的值在积分过程的递归图.

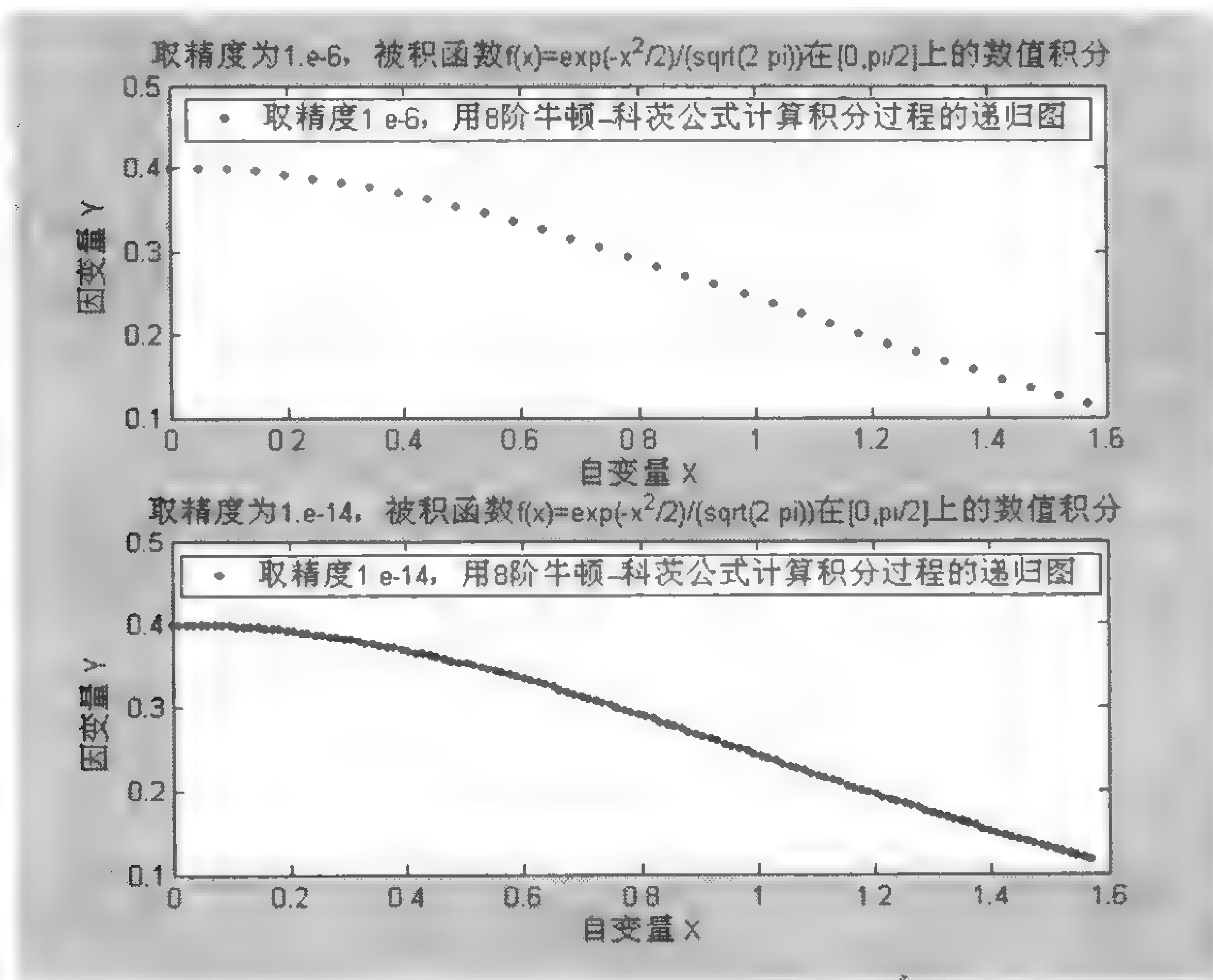


图 9-4 用 quad8 计算 I 的过程中被积函数 $f(x)=\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$ 在积分过程的递归图

例 9.6.8 用辛普森求积公式、8 阶牛顿-科茨求积公式和洛巴托求积公式分别计算 $I=\int_0^{\frac{\pi}{2}} e^{x^3}\cos 2x\mathrm{d}x$，取精度分别为 10^{-14} 和 10^{-4}，并与精确值比较. 请再改用取 10 个节点，分别用三阶拉格朗日插值多项式和三次样条构造被积函数，然后用辛普森求积公式、8 阶牛顿-科茨求积公式和洛巴托求积公式分别计算 I，取计算精度分别为 10^{-4}. 再改用 100，1 000，10 000 个节点，用三次样条构造被积函数计算，观察计算结果与精确值的绝对误差有何变化.

解 (1) 建立名为 fun.m 的 M 文件

```
function y = fun(x)
y = exp(x.^3).*cos(2*x);
```

取精度为 10^{-4}，输入程序

```
>> F = inline('exp(x.^3).*cos(2*x)'); [Q1,FCNT1] = quadl
(@fun,0,pi/2,1.e-4,3)
```

```
[Q8,FCNT8] = quad8(@ fun,0,pi/2,1.e-4,3)
[Q,FCNT] = quad(@ fun,0,pi/2,1.e-4,3)
syms x
fi = int(exp(x.^3).*cos(2*x),x,0, pi/2);
Fs = double(fi), wQ1 = double(abs(fi-Q1))
wQ8 = double(abs(fi-Q8)), wQ = double(abs(fi-Q))
```

运行后屏幕显示 I 的精确值的近似值 F_s,取精度为 10^{-4},分别用函数 quadl 和 quad8 及 quad 分别计算 I 的近似值 Q_1,Q_8和 Q,递归次数 $FCNT1$,$FCNT8$ 和 $FCNT$,近似值分别与精确值f_i的绝对误差 wQ_1,wQ_8和 wQ 如下

```
Q1 =
   -6.45505792320975
FCNT1 =
    48
Q8 =
   -6.45505883179317
FCNT8 =
    33
Q =
   -6.45506213501363
FCNT =
    33
Warning: Explicit integral could not be found.
 > In C:\MATLAB6p5\toolbox\symbolic\@ sym\int.m at line 58
Fs =
   -6.45505792116786
wQ1 =
    2.041896319084332e-009
wQ8 =
    9.106253137107240e-007
wQ =
    4.213845774426019e-006
```

由此可见,用洛巴托求积公式,递归 48 次,计算数值积分的精度 wQ_1最高,用 8 阶牛顿-科茨求积公式,递归 33 次,计算数值积分的精度 wQ_8次之,用辛普森求积公式,递归 33 次,计算数值积分的精度 wQ 最低. 这三种方法的绝对误差都小于精度 10^{-4},已经达到给定的精度.

(2) 取精度为 10^{-14},输入程序

```
>> F = inline('exp(x.^3).*cos(2*x)');
[Q1,FCNT1] = quadl(@ fun,0,pi/2,1.e-14,3)
```

```
[Q8,FCNT8] = quad8(@ fun,0,pi/2,1.e - 14,3)
[Q,FCNT] = quad(@ fun,0,pi/2,1.e - 14,3)
syms x
fi = int(exp(x.^3).* cos(2 * x),x,0, pi/2);
Fs = double(fi), wQ1 = double(abs(fi - Q1))
wQ8 = double(abs(fi - Q8)), wQ = double(abs(fi - Q))
```

运行后屏幕显示 I 的精确值的近似值 Fs,取精度为 10^{-14},分别用函数 quadl 和 quad8 及 quad 分别计算 I 的近似值 Q_1,Q_8和 Q,递归次数 $FCNT1$,$FCNT8$ 和 $FCNT$,近似值分别与精确值f_i的绝对误差 wQ_1,wQ_8和 wQ 如下

```
Q1  =
  -6.45505792116785
FCNT1  =
  798
Q8  =
  -6.45505792116785
FCNT8  =
        4273
Q  =
  -6.45505792116785
FCNT  =
        2961
Warning: Explicit integral could not be found.
 > In C:\MATLAB6p5\toolbox\symbolic\@ sym\int.m at line 58
Fs  =
  -6.45505792116786
wQ1  =
    3.663345763769995e-015
wQ8  =
    1.886988924369744e-015
wQ  =
    2.775167344069869e-015
```

这三种方法计算结果的绝对误差都小于 10^{-14},达到所要求的精度.

(3) 取 10 个节点,改用三阶拉格朗日插值多项式构造被积函数,然后用辛普森求积公式、8 阶牛顿－科茨求积公式和洛巴托求积公式分别计算 I,取精度分别为 10^{-14},输入程序

```
>> X = [0,0.1745,0.3491,0.5236,0.6981,0.8727,1.0472,1.2217,
1.3963,1.5708];
Y = exp(X.^3).* cos(2 * X); L1 = lfun(X(1:4),Y(1:4)),
```

```
L2 = lfun(X(4:7),Y(4:7)), L3 = lfun(X(7:10),Y(7:10)),
```

运行后输出多项式 L_1, L_2, L_3. 然后输入程序

```
>> syms x
L1 = inline('
528475254238581253770784020141/1267650600228229401496703205376 * x.^3 -21
509217334629805675493576414 95/1267650600228229401496703205376 * x.^2 -110
805035875068143559156492912389/3318709271397504573118368991674368 * x +1 ');
L2 = inline('
-325145953093990853355298421155 99/2535301200456458802993406410752 * x.^3
+2888077768358996481638550437 0251/1267650600228229401496703205376 * x.^2
-9690424145215092365005571145155/633825300114114700748351602688 * x +2118
127727532097185721358364133 9/5070602400912917605986812821504 ');
L3 = inline('
-223476575673079066859855924604 79/39614081257132168796771975168 * x.^3 +1
943976462782510599598993738 3455/99035203142830421991929937 92 * x.^2 -4534
685551911754756140528136570 7/19807040628566084398385987584 * x +17651524
197374204698139309247929/19807040628566084398385987584 ');
[Q1,FCNT1] = quad(L1,0, 0.5236,1.e-4);
[Q2,FCNT2] =quad(L2, 0.5236, 1.0472,1.e-4);
[Q3,FCNT3] - quad(L3, 1.0472, pi/2,1.e-4);
Q = Q1 + Q2 + Q3 ,FCNT = FCNT1 + FCNT2 + FCNT3
[Q81,FCNT81] = quad8(L1,0, 0.5236,1.e-4);
[Q82,FCNT82] =quad8(L2, 0.5236, 1.0472,1.e-4);
[Q83,FCNT83] = quad8(L3, 1.0472, pi/2,1.e-4);
Q8 = Q81 + Q82 + Q83 ,FCN8T = FCNT81 + FCNT82 + FCNT83
[Q11,FCNT11] = quadl(L1,0, 0.5236,1.e-4);
[Q12,FCNT12] =quadl(L2, 0.5236, 1.0472,1.e-4);
[Q13,FCNT13] = quadl(L3, 1.0472, pi/2,1.e-4);
Ql = Q11 + Q12 + Q13 ,FCNTl = FCNT11 + FCNT12 + FCNT13
syms x
fi =int(exp(x.^3).*cos(2*x),x,0, pi/2);
Fs = double (fi), wQl = double (abs(fi -Ql) )
wQ8 - double (abs(fi -Q8) ), wQ = double (abs(fi -Q) )
```

运行后屏幕显示首先用三阶拉格朗日插值多项式构造被积函数,然后用辛普森求积公式、8 阶牛顿 - 科茨求积公式和洛巴托求积公式分别计算 I 的近似值 Q_1, Q_8和 Q,递归次数 $FCNT1$, $FCNT8$ 和 $FCNT$,近似值分别与精确值f_i的绝对误差 wQ_1, wQ_8和 wQ 如下

```
Q =                          FCNT =     Q8  =                    FCN8T  =
  -6.63556515894584   39           -6.63556515894593          99
Q1 =                          FCNT1 =
   -6.63556515894607           54
Warning: Explicit integral could not be found.
> In D:\MATLAB6P5\toolbox\symbolic\@ sym\int.m at line 58
Fs =                                    wQ1 =
     -6.45505792116786              0.18050723777821
wQ8 =                                   wQ =
     0.18050723777807               0.18050723777798
```

虽然三种计算的结果都没有达到要求精度 10^{-4}，但是它们计算结果都约等于 -6.635 565 158 900 00 与精确值 -6.455 057 921 167 86…很接近. 如果增加节点的个数，就能够达到给定的精度（请读者仿效此例完成）.

(4) 取 10 个节点，改用三次样条构造被积函数，然后用辛普森求积公式、8 阶牛顿-科茨求积公式和洛巴托求积公式分别计算 I，取精度分别为 10^{-4}.

首先建立并保存名为 scjfQ18.m 的 M 文件

```
function wQ = scjfQ18(n)
a =0; b = pi/2;X =0:pi/(2*n-1):pi/2; fs = exp(X.^3).*cos(2* X);
pp = spline(X,fs); Qs1 = quadl(@ ppval,a,b,[],[],pp)
Qs8 = quad8(@ ppval,a,b,[],[],pp),
Qs - quad(@ ppval,a,b,[],[],pp)
syms x
fi = int(exp(x.^3).*cos(2*x),x,0, pi/2); Fs = double(fi)
wQ1 = double(abs(fi - Qs1)),
wQ8 = double(abs(fi - Qs8))
wQ = double(abs(fi - Qs))
```

然后输入程序

```
>> n =10;wQ = scjfQ18(n)
```

运行后屏幕显示结果如下

```
Qs1 =                  Qs8 =                   Qs  =
  -6.33658113206801  -6.33658632890273   -6.33659053214403
Warning: Explicit integral could not be found.
> In C:\MATLAB6p5\toolbox\symbolic\@ sym\int.m at line 58
  In C:\MATLAB6p5\work\scjfQ18.m at line 7
Fs =
   -6.45505792116786
wQ1 =                   wQ8 =                   wQ =
```

```
        0.11847678909985    0.11847159226513    0.11846738902383
```

虽然三种计算的结果都没有达到要求精度,但是它们计算结果都约等于 -6.336 6与精确值 -6.455 057 921 167 8…很接近,其绝对误差已经约减少到 0.118 5,比用二阶拉格朗日插值多项式构造被积函数计算的误差还小.

(5) 如果节点数增加到100, 1 000, 10 000 个,用三次样条构造被积函数计算,计算结果和精确值及其绝对误差如下表 9-6 所示:

表 9-6 节点数 n = 10, 100, 1 000, 10 000,用三次样条构造被积函数计算的结果

	quad1	
n	近似值	绝对误差
10	-6.336 581 132 1	1.184 767 89e-001
100	-6.455 054 760 3	3.160 878 71e-006
1 000	-6.455 057 922 0	7.520 685 19e-010
10 000	-6.455 057 923 2	2.041 762 20e-009
	quad8	
n	近似值	绝对误差
10	-6.336 586 328 9	1.184 715 92e-001
100	-6.455 023 632 1	3.428 907 30e-005
1 000	-6.455 058 827 5	9.063 243 78e-007
10 000	-6.455 058 831 8	9.106 248 99e-007
	quad	
n	近似值	绝对误差
10	-6.336 590 532 1	1.184 673 89e-001
100	-6.455 052 169 0	5.752 139 57e-006
1 000	-6.455 057 977 0	5.587 255 62e-008
10 000	-6.455 057 977 9	5.671 284 59e-008

从表 9-6 可见,随着节点个数 n 的增加,定积分的近似值与精确值的绝对误差在不断地减少,当 n = 10 000 时,分别用函数 quad1,quad8,quad 计算的定积分的近似值与精确值的绝对误差已经分别达到 wQ_1 = 2.041 762 20e-009, wQ_8 = 9.106 248 99e-007, wQ = 5.671 284 59e-008. 因此,选择合适的节点的

个数 n,就能够达到要求的精度(请读者仿效此例完成).

例 9.6.9 用 MATLAB 函数和辛普森求积公式、8 阶牛顿－科茨求积公式和洛巴托求积公式分别计算无界函数的反常积分 $\int_0^1 \frac{1}{\sqrt{x}}dx$,并与精确值比较.

解 首先将下面名为 fun1.m 的程序保存为 M 文件

```
function y = fun1(x)
          y =x.^( -1/2);
```

然后在 MATLAB 工作窗口输入程序

```
>>[Ql ,FCNTl] = quadl(@ fun1,0,1,1.e-14,3)
  [Q ,FCNT] = quad(@ fun1,0,1,1.e-14,3)
syms x, fi =int(x.^( -1/2),x,0,1), wQl = double (abs(fi -Ql) )
wQ = double (abs( fi - Q) ), [Q8 ,FCNT8] = quad8 (@ fun1,0,1,
1.e-14,3)
wQ8 = double (abs( fi -Q8) )
```

运行后屏幕以滚动形式显示在分别用函数 quadl 和 quad 分别递归求积分的整个过程的[fcnt a b-a Q]的值,计算 I 的近似值 Q_1 和 Q,递归次数 $FCNT1$ 和 $FCNT$,近似值分别与精确值 f_i 的绝对误差 wQ_1 和 wQ 如下

```
Warning: Minimum step size reached; singularity possible.
(Type "warning off MATLAB:quadl:MinStepSize" to suppress this
warning.)
Ql =                        FCNTl =
     1.99999999937036             5089
  Warning: Minimum step size reached; singularity possible.
(Type "warning off MATLAB:quad:MinStepSize" to suppress this
warning.)
Q =                 FCNT =       fi =       wQl =
  1.99999999979330     6146            2      6.296390075988256e-010
wQ =
   2.066991022786624e-010
```

可见,用函数 quad8 不能计算此无界函数的反常积分.

例 9.6.10 测得定积分 $I=\int_0^{\frac{\pi}{2}} f(x)\,dx$ 中被积函数 $Y=f(x)$ 在积分变量 x 的某几个特定 $X=0$, 0.157 1, 0.471 2, 0.549 8, 0.628 3, 0.863 9, 1.021 0, 1.099 6, 1.570 9 处的值分别为 $Y=0$, 0.156 5, 0.454 0, 0.522 5, 0.587 8, 0.760 4, 0.852 6, 0.891 0, 1.000 0,试根据这些值首先分别用分段二、三阶拉格朗日插值多项式和三次样条插值构造被积函数,然后分别用辛普森和洛巴托求积公式计算,取精度分别为 10^{-14} 和 10^{-4},并与精确值 1 进行比较.

解 下面分别用分段二阶、三阶拉格朗日插值多项式和三次样条插值方法分别构造被积函数,计算数值积分.

方法 1 用二阶拉格朗日插值多项式构造被积函数,输入程序

```
>> X =[0,0.1571, 0.4712,0.5498,0.6283,0.8639, 1.0210,1.0996,
1.5708];
Y = [0,0.1565,0.4540,0.5225,0.5878,0.7604,0.8526,0.8910,
1.0000];
L1 =lfun(X(1:3),Y(1:3)), L2 =lfun(X(3:5),Y(3:5)),
L3 =lfun(X(5:7),Y(5:7)), L4 =lfun(X(7:9),Y(7:9)),
```

运行后输出多项式 L_1,L_2,L_3,L_4,然后输入程序

```
>> syms x
L1 = inline('-1464428152621 4207/140737488355328000 * x.^2 +
2280009553133670687/2251799813685248000 * x');
L2 = inline('-88810055001957943/351843720888320000 * x.^2 +
15892292661979563/14073748835532800 * x -15511422992738729/
703687441776640000');
L3 = inline('-261101353037957127/703687441776640000 * x.^2 +
362054074498830351/281474976710656000 * x -207985426949036897/
2814749767106560000');
L4 = inline('-131688789655946847/281474976710656000 * x.^2 +
104193434945799191/70368744177664000 * x -930974059276412576 8759/
54295819320043765760000');
[Q141,FCNT141] = quad(L1,0, 0.4712,1.e-14);
[Q142,FCNT142] =quad(L2, 0.4712, 0.6283,1.e-14);
[Q143,FCNT143] = quad(L3, 0.6283, 1.0210,1.e-14);
[Q144,FCNT144] - quad(L4, 1.0210, pi/2,1.e 14);
Q14 = Q141 + Q142 + Q143 + Q144,
FCNT14 = FCNT141 + FCNT142 + FCNT143 + FCNT144
[Q41,FCNT41] = quad(L1,0, 0.4712,1.e-4);
[Q42,FCNT42] =quad(L2, 0.4712, 0.6283,1.e-4);
[Q43,FCNT43] = quad(L3, 0.6283, 1.0210,1.e-4);
[Q44,FCNT44] = quad(L4, 1.0210, pi/2,1.e-4);
Q4 = Q41 + Q42 + Q43 + Q44,
FCNT4 = FCNT41 + FCNT42 + FCNT43 + FCNT44
[Q141,FCNT141] = quadl(L1,0, 0.4712,1.e-4);
[Q142,FCNT142] =quadl(L2, 0.4712, 0.6283,1.e-4);
[Q143,FCNT143] = quadl(L3, 0.6283, 1.0210,1.e-4);
[Q144,FCNT144] = quadl(L4, 1.0210, pi/2,1.e-4);
```

```
Q14 = Q141 + Q142 + Q143 + Q144,
FCNT14 = FCNT141 + FCNT142 + FCNT143 + FCNT144
[Q1141,FCNT1141] = quadl(L1,0, 0.4712,1.e-14,2);
[Q1142,FCNT1142] =quadl(L2, 0.4712, 0.6283,1.e-14,2);
[Q1143,FCNT1143] = quadl(L3, 0.6283, 1.0210,1.e-14,2);
[Q1144,FCNT1144] = quadl(L4, 1.0210, pi/2,1.e-14,2);
Q114 = Q1141 + Q1142 + Q1143 + Q1144,
FCNT114 = FCNT1141 + FCNT1142 + FCNT1143 + FCNT1144
wQ14 =double(abs(1 - Q14)), wQ4 =double(abs(1 - Q4))
wQ14 =double(abs(1 - Q14)), wQ114 =double(abs(1 - Q114))
```

运行后屏幕显示如下

```
Q14 =                    FCNT14 =     Q4 =                    FCNT4 =
   0.99957595925413          52          0.99957595925413         52
Q14 =                    FCNT14 =     Q114 =                  FCNT114 =
   0.99957595925413          72          0.99957595925413         72
wQ14 =                                wQ4 =
   4.240407458748763e-004                4.240407458748763e-004
wQ14 =                                wQ114 =
   4.240407458748763e-004                4.240407458748763e-004
```

方法 2 用三阶拉格朗日插值多项式构造被积函数,输入程序

```
>>  X =[0,0.1571, 0.4712,0.5498,0.6283,0.8639, 1.0210,1.0996,
1.5708];
 Y = [0, 0.1565, 0.4540, 0.5225, 0.5878, 0.7604, 0.8526, 0.8910,
 1.0000];
L1 =lfun(X(1:4),Y(1:4)), L2 =lfun(X(4:7),Y(4:7)),
 L3 =lfun(X(7:9),Y(7:9)),
```

运行后输出多项式 L_1, L_2, L_3,然后输入程序

```
>> syms x
L1 = inline('-9070382563990323/5629499534213120 0 * x.^3 -
793956227651389/281474976710656000 * x.^2 +2253151961697736691/
2251799813685248000 * x');
L2 = inline('-41118170146065463/351843720888320000 * x.^3 -
1700780706742359/21990232555520000 * x.^2 +735208215754549839/
703687441776640000 * x -25688208496779769/2814749767106560000');
L3 = inline('-1316887896559468 47/281474976710656000 * x.^2 +
1041934349457991 91/70368744177664000 * x -9309740592764125768759/
5429581932004376576 0000');
[Q141,FCNT141] = quad(L1,0, 0.5498,1.e-14);
```

```
[Q142,FCNT142] =quad(L2, 0.5498, 1.0210,1.e-14);
[Q143,FCNT143] = quad(L3, 1.0210, pi/2,1.e-14);
Q14 = Q141 + Q142 + Q143 ,FCNT14 = FCNT141 + FCNT142 + FCNT143
[Q41,FCNT41] = quad(L1,0, 0.5498,1.e-4);
[Q42,FCNT42] =quad(L2, 0.5498, 1.0210,1.e-4);
[Q43,FCNT43] = quad(L3, 1.0210, pi/2,1.e-4);
Q4 = Q41 + Q42 + Q43, FCNT4 - FCNT41 + FCNT42 + FCNT43
[Q141,FCNT141] = quadl(L1,0, 0.5498,1.e-14);
[Q142,FCNT142] =quadl(L2, 0.5498, 1.0210,1.e-14);
[Q143,FCNT143] = quadl(L3, 1.0210, pi/2,1.e-4);
Q14 = Q141 + Q142 + Q143, FCNT14 = FCNT141 + FCNT142 + FCNT143
[Q1141,FCNT1141] = quadl(L1,0, 0.5498,1.e-14);
[Q1142,FCNT1142] =quadl(L2, 0.5498, 1.0210,1.e-14);
[Q1143,FCNT1143] = quadl(L3, 1.0210, pi/2,1.e-4);
Q114 = Q1141 + Q1142 + Q1143, FCNT114 = FCNT1141 + FCNT1142 +
FCNT1143
wQ14 =double(abs(1 - Q14)), wQ4 =double(abs(1 - Q4))
wQ14 =double(abs(1 - Q14)), wQ114 =double(abs(1 - Q114))
```

运行后屏幕显示如下

```
Q14 =                    FCNT14 -   Q4 -                  FCNT4 =
   0.99975250938584          39       0.99975250938584          39
Q14 =                     FCNT14 =  Q114 =                 FCNT114 =
   0.99975250938584          54       0.99975250938584          54
wQ14 =                             wQ4 =
   2.474906141644029e-004             2.474906141644029e-004
wQ14 =                             wQ114 =
   2.474906141642919e-004             2.474906141642919e-004
```

方法 3 三次样条插值多项式计算数值积分.

输入程序

```
>> a=0; b= pi/2;
X=[0,0.1571, 0.4712,0.5498,0.6283,0.8639, 1.0210,1.0996,1.5708];
fs=[0,0.1565,0.4540,0.5225,0.5878,0.7604,0.8526,0.8910,1.0000];
pp=spline(X,fs);
Qs1=quadl(@ ppval,a,b,[],[],pp),wQs1 =double(abs(1-Qs1))
Qs8=quad8(@ ppval,a,b,[],[],pp),wQs8 =double(abs(1-Qs8))
Qs=quad(@ ppval,a,b,[],[],pp),wQs =double(abs(1-Qs))
```

运行后屏幕显示结果如下

```
Qs1 =                      Qs8 =                    Qs =
```

```
     1.00012614786269        1.00012598484705      1.00012589347894
   wQs1 =                wQs8 =                   wQs =
1.261478626870538e-004  1.259848470498426e-004  1.258934789407640e-004
```

由上面的三种方法可以看出:在不等距节点的情况下,用样条插值构造被积函数,然后用自适应辛普森公式计算的误差 $wQs=1.258\,934\,789\,407\,640\text{e}-004$ 最小,用洛巴托求积公式计算的误差 $wQs_1=1.261\,478\,626\,870\,538\text{e}-004$ 也很小;用三阶拉格朗日插值多项式构造被积函数比用二阶拉格朗日插值多项式构造被积函数计算的误差小,并且计算时取的精度 10^{-14} 和 10^{-4} 对计算结果的误差影响不大.用上面的三种方法计算数值积分的结果与精确值1很接近,有实际意义.

例 9.6.11 在某次实验中测得一质点在某几个特定时间 t 的速度 $v=v(t)$ 被列入下表,首先根据下表中给定的数据分别用分段二阶、三阶拉格朗日插值多项式、样条插值构造被积函数,然后用分别用洛巴托求积公式计算在这段时间内质点的位移,并与精确值 $0.2(e^4-e)\approx 10.375\,973\,640\,937\,04$ 比较.

$t(s)$	1.000 0	1.500 0	2.000 0	2.500 0	3.000 0	3.500 0	4.000 0
$v(t)(m/s)$	0.543 7	0.896 3	1.477 8	2.436 5	4.017 1	6.623 1	10.919 6

解 因为所求在这段时间 [1.000 0,4.000 0] 内质点的位移为 $s=\int_1^4 v(t)\,\mathrm{d}t$,所以下面分别用分段二阶、三阶拉格朗日插值多项式、分段埃尔米特插值、样条插值这四种方法分别构造被积函数,计算数值积分.

方法 1 用二阶拉格朗日插值多项式构造被积函数,输入程序

```
>> t=[1.000,1.5000,2.0000,2.5000,3.0000,3.5000,4.0000];
v=[0.5437,0.8963,1.4778,2.4365,4.0171,6.6231,10.9196];
L1=lfun(t(1:3),v(1:3)), L2=lfun(t(3:5),v(3:5)),
L3=lfun(t(5:7),v(5:7)),
```

运行后输出多项式 L_1,L_2,L_3,然后输入程序

```
>> syms x
L1 = inline('2289/5000*x.^2-4393/10000*x+1313/2500');
L2 = inline('6219/5000*x.^2-36797/10000*x+1931/500');
L3 = inline('3381/1000*x.^2-33529/2000*x+14926/625');
[Q141,FCNT141] = quadl(L1,1,2.0000,1.e-14);
[Q142,FCNT142] =quadl(L2,2.0000,3.0000,1.e-14);
[Q143,FCNT143] = quadl(L3,3.0000,4.0000,1.e-14);
Q14 = Q141 + Q142 + Q143, FCNT14 = FCNT141 + FCNT142 + FCNT143
[Q41,FCNT41] = quadl(L1,1,2.0000,1.e-4);
```

```
[Q42,FCNT42] = quadl(L2, 2.0000, 3.0000,1.e-4);
[Q43,FCNT43] = quadl(L3, 3.0000, 4.0000,1.e-4);
Q4 = Q41 + Q42 + Q43, FCNT4 = FCNT41 + FCNT42 + FCNT43
wQ14 =double(abs(10.37597364093704 - Q14))
wQ4 -double(abs(10.37597364093704 - Q4))
```

运行后屏幕显示用分段二阶拉格朗日插值多项式构造被积函数,然后用洛巴托求积公式计算在这段时间内质点的位移的近似值 Q_{14}, Q_4 和迭代次数 *FCNT*14, *FCNT*4,取精度分别为 10^{-14} 和 10^{-4} 如下

```
Q14 =                  FCNT14 =     Q4 =                FCNT4 =
   10.37944999999999       54        10.37944999999999      54
wQ14 =                 wQ4 =
   0.00347635906295      0.00347635906295
```

方法 2 用三阶拉格朗日插值多项式构造被积函数,输入程序

```
>> t =[1.000, 1.5000, 2.0000, 2.5000, 3.0000, 3.5000, 4.0000];
v =[ 0.5437,0.8963,1.4778,2.4365,4.0171,6.6231,10.9196];
L1 =lfun(t(1:4),v(1:4)), L2 =lfun(t(4:7),v(4:7)),
```

运行后输出多项式 L_1, L_2,然后输入程序

```
>> syms x
L1 = inline('1483/7500*x.^3 -54/125*x.^2 +25379/30000*x -
17/250');
L2 = inline('2217/2500*x.^3 -7413/1250*x.^2 +156037/10000
*x -3341/250');
[Q141,FCNT141] = quadl(L1,1, 2.5000,1.e-14);
[Q142,FCNT142] =quadl(L2, 2.5000, 4.0000,1.e-14);
Q14 = Q141 + Q142
FCNT14 = FCNT141 + FCNT142
[Q41,FCNT41] = quadl(L1,1, 2.5000,1.e-4);
[Q42,FCNT42] =quadl(L2, 2.5000, 4.0000,1.e-4);
Q4 = Q41 + Q42,
FCNT4 - FCNT41 + FCNT42
wQ14 =double(abs(10.37597364093704 - Q14))
wQ4 =double(abs(10.37597364093704 - Q4))
```

运行后屏幕显示用分段三阶拉格朗日插值多项式构造被积函数,然后用洛巴托求积公式计算在这段时间内质点的位移的近似值 Q_{14}, Q_4 和迭代次数 *FCNT*14, *FCNT*4,取精度分别为 10^{-14} 和 10^{-4} 如下

```
Q14 =                  FCNT14 =      Q4 =                 FCNT4 =
   10.38360000000001        36       10.38360000000001       36
```

```
wQ14 =                  wQ4 =
    0.00762635906297        0.00762635906297
```

方法 3 三次样条插值多项式计算数值积分.

输入程序

```
>> a=1; b=4;t=[1.000, 1.5000, 2.0000, 2.5000, 3.0000, 3.5000,
4.0000];
v=[ 0.5437,0.8963,1.4778,2.4365,4.0171,6.6231,10.9196];
pp=spline(t,v); Qsl=quadl(@ ppval,a,b,[],[],pp)
Qs8=quad8(@ ppval,a,b,[],[],pp)
Qs=quad(@ ppval,a,b,[],[],pp)
wQsl=double(abs(10.37597364093704 -Qsl))
wQs8=double(abs(10.37597364093704 -Qs8))
wQs=double(abs(10.37597364093704 -Qs))
```

运行后屏幕显示用分段样条插值构造被积函数,然后用洛巴托求积公式计算在这段时间内质点的位移的近似值 Q_{14},Q_4和迭代次数 *FCNT*14,*FCNT*4,取精度分别为 10^{-14}和 10^{-4}如下

```
Qsl =                       Qs8 =
    10.37862711220329           10.37863802976191
Qs =                        wQsl =
    10.37862431698066           0.00265347126625
wQs8 =                      wQs =
    0.00266438882486            0.00265067604361
```

在等距节点的情况下,由上面的三种方法可以看出:用三次样条插值构造被积函数,计算数值积分的结果与精确值的误差约为 0.002 7 最小;其次是用二阶拉格朗日插值多项式构造被积函数,然后用洛巴托求积公式计算在这段时间内质点的位移的近似值(递归 54 次)与精确值的绝对误差约为 0.003 5;最后是用三阶拉格朗日插值多项式构造被积函数,递归 36 次,计算数值积分的结果与精确值的绝对误差约为 0.007 6,与精确值很接近,三种方法计算结果有实际意义.

例 9.6.12(卫星轨道长度) 人造地球卫星轨道可视为平面上的椭圆.我国第一颗人造地球卫星近地点距地球表面 439 km,远地点距地球表面 2 384 km,地球半径为 6 371 km,求该卫星的轨道长度.

解 卫星轨道椭圆的参数方程为 $x=a\cos t, y=b\sin t\ (0\leqslant t\leqslant 2\pi)$,$a,b$ 分别是长、短半轴.根据计算参数方程的弧长的公式,椭圆长度可表为如下积分

$$L=4\int_0^{\frac{\pi}{2}}(a^2\sin^2 t+b^2\cos^2 t)^{1/2}\mathrm{d}t,$$

称椭圆积分,它无法用解析方法计算. 根据所给数据 $a = 6\ 371 + 2\ 384 = 8\ 755$, $b = 6\ 371 + 439 = 6\ 810$. 下面是用梯形公式、辛普森公式和洛巴托求积公式计算的具体方法

在 MATLAB 工作区输入程序

```
>> t =0:pi/10:pi/2; a =8755; b =6810;
y = sqrt(a^2 * sin(t).^2 + b^2 * cos(t).^2); l1 =4 * trapz(t,y)
syms t
Y = inline('sqrt(8755^2 * sin(t).^2 +6810^2 * cos(t).^2)');
[Qq1,FCNT1] = quadl(Y,0,pi/2,1e -6)
Q1 =4 * Qq1, [Qq,FCNT] = quad(Y,0,pi/2,1e -6), Q =4 * Qq
```

输出结果为

```
l1 =            FCNT1 =    Q1 =             FCNT =     Q =
  4.9090e+004      48       4.9090e+004       109      4.9090e+004
```

可以看出,用梯形公式、辛普森公式和洛巴托求积公式计算的结果相差无几,且梯形公式(仅将区间 5 等分)给出了很好的结果. 轨道长度为 4.909×10^4 km.



习 题 9.6

1. 用自适应积分计算定积分 $I = \int_0^{\frac{\pi}{2}} e^{\sin x} dx$,取精度 10^{-4},作出它们的积分图,并与精确值进行比较,估计误差.

2. 分别用三点高斯 - 勒让德积分公式、步长为 3 的梯形公式和步长为 2 的辛普森求积公式计算 $\int_{-1}^{1} \frac{1}{5+x} dx$,并将计算结果与精确值进行比较.

3. 用五点高斯 - 勒让德积分公式计算 $\int_0^{1.5} \frac{\sin x}{1+2x} dx$,取精度为 10^{-7}.

4. 用七点高斯 - 勒让德积分公式计算 $I = \frac{1}{\sqrt{2\pi}} \int_0^1 e^{-\frac{x^2}{2}} dx$,并将计算结果与精确值比较.

5. 利用一般型高斯 - 勒让德积分公式计算 $I = \frac{1}{\sqrt{2\pi}} \int_0^2 e^{-\frac{x^2}{2}} dx$,取代数精度为 13,并与精确值比较.

6. 在某次实验中测得一质点在某几个特定时间 t 的速度 $v = v(t)$ 被列入下表,首先根据下表中给定的数据分别用分段三阶埃尔米特插值、分段二阶、三阶拉格朗日插值多项式、三次样条插值构造被积函数,然后用分别用洛巴托求积公式计算在这段时间内质点的位移,并与精确值 $0.2(e^4 - e) \approx 10.375\ 973\ 640\ 937\ 04$ 比较.

t(s)	1.000 0	1.500 0	2.000 0	2.500 0	3.000 0	3.500 0	4.000 0
$v(t)$(m/s)	0.543 7	0.896 3	1.477 8	2.436 5	4.017 1	6.623 1	10.919 6

7. 用 MATLAB 函数和辛普森求积公式、8 阶牛顿－科茨求积公式和洛巴托求积公式分别计算无界函数的反常积分 $\int_0^1 \frac{1}{\sqrt{x}}dx$，并与精确值比较.

8. 用辛普森求积公式、8 阶牛顿－科茨求积公式和洛巴托求积公式分别计算 $I=\int_0^{\frac{\pi}{2}} e^{x^3}\cos 3x dx$，取精度分别为 10^{-14} 和 10^{-4}，并与精确值比较.

9. 用函数 quadl 和 quad8 及 quad 分别数值计算 $I=\frac{1}{\sqrt{2\pi}}\int_0^{+\infty} e^{-\frac{x^2}{2}}dx$，分别取精度为 10^{-6} 和 10^{-14}，将计算结果分别与精确值比较，并画出图形.

10. 用函数 quadl 和 quad8 及 quad 分别数值计算 $I=\int_0^7 e^{-\frac{x^2}{2}}dx$，分别取精度为 10^{-6} 和 10^{-14}，将计算结果分别与精确值比较，并画出图形.

11. 用辛普森求积公式、8 阶牛顿－科茨求积公式和洛巴托求积公式分别计算 $I=\int_0^{2\pi} e^{x^3}\cos 2x dx$ 和 $I=\int_0^{\frac{\pi}{2}} e^{x^3}\cos 2x dx$，取精度分别为 10^{-14} 和 10^{-4}，并与精确值比较. 如果计算结果严重失真，请改用取 10 个节点，分别用三阶拉格朗日插值多项式和三次样条构造被积函数，然后用辛普森求积公式、8 阶牛顿－科茨求积公式和洛巴托求积公式分别计算 I，取计算精度分别为 10^{-4}. 再改用 100, 1 000, 10 000 个节点，用三次样条构造被积函数计算，观察计算结果与精确值的绝对误差有何变化.

12. 用矩形、梯形、高斯－勒让德积分公式和辛普森四种求积公式计算由下表数据给出的积分 $I=\int_{0.3}^{1.5} y(x)dx$，并将计算值与精确值作比较.

k	1	2	3	4	5	6	7
x_k	0.3	0.5	0.7	0.9	1.1	1.3	1.5
y_k	0.389 5	0.659 8	0.914 7	1.161 1	1.397 1	1.621 2	1.832 5

13. 选择一些函数用梯形、辛普森求积公式、8 阶牛顿－科茨求积公式、洛巴托求积公式、高斯－勒让德积分公式、利用插值方法分别计算数值积分. 改变步长（对梯形），改变精度要求（对辛普森），改变节点数目，进行比较、分析. 如下函数供选择参考：

(1) $y=\frac{2}{x^2+1}$, $0\leqslant x\leqslant 1$；　(2) $y=e^{-5x}\cos 3x$, $0\leqslant x\leqslant 2$；

(3) $y=\sqrt{5+2x^2}$, $0\leqslant x\leqslant 2$；　(4) $y=\frac{1}{5\sqrt{2\pi}}e^{-\frac{(x-1)^2}{10}}$, $-2\leqslant x\leqslant 3$.

14. 用数值积分方法选六种途径计算 π.

15. 在节点数目同样多的条件下，分别用精度为 10^{-10} 的辛普森求积公式和代数精度为

15 的高斯－勒让德求积公式计算 $I=\int_0^{\frac{\pi}{2}}\sin x\mathrm{d}x$，并将计算值与精确值作比较.

16. 弹簧在周期性外力作用下（忽略阻力）其位移 $x(t)$ 满足微分方程 $mx''+kx=F\cos\omega t$，在初始条件 $x(0)=x'(0)=0$ 下的解为

$$x(t)=\frac{2F}{m(\omega_0^2-\omega^2)}\sin\frac{\omega_0-\omega}{2}t\sin\frac{\omega_0+\omega}{2}t,\ \omega_0=\sqrt{\frac{k}{m}}\neq\omega.$$

已知 $m=1,k=9,F=1,\omega=2$，求 $0\leqslant t\leqslant 2$ 内 $x(t)$ 的平均值，作 $x(t)$ 和这个平均值的图形.

17. 确定系数 A_{-1},A_0,A_1，使求积公式

$$\int_{-h}^{h}f(x)\mathrm{d}x\approx A_{-1}f(-h)+A_0f(0)+A_1f(h)$$

具有尽可能高的代数精度，并指出所得求积公式的代数精度.

9.7 反常积分的计算及其 MATLAB 程序

积分区间为无穷区间，或被积函数为无界函数的积分，称之为反常积分. 本节主要介绍反常积分的符号计算和数值计算及其 MATLAB 程序.

9.7.1 无穷积分的符号计算及其 MATLAB 程序

无穷区间上的反常积分的符号计算可以用计算机软件 MATLAB 系统提供的函数 int 来实现. 下面通过实例介绍用 MATLAB 软件计算反常积分的方法.

无穷区间上的反常积分有三种如下基本类型：

(1) $\int_a^{+\infty}f(x)\mathrm{d}x$ 可以用 MATLAB 程序 `F1 = int(f(x),x,a,+inf)` 计算.

若 $\lim\limits_{b\to+\infty}\int_a^b f(x)\mathrm{d}x$ 存在，则称反常积分 $\int_a^{+\infty}f(x)\mathrm{d}x$ 存在或收敛，且 $\int_a^{+\infty}f(x)\mathrm{d}x=\lim\limits_{b\to+\infty}\int_a^b f(x)\mathrm{d}x$，否则称反常积分不存在或发散.

(2) $\int_{-\infty}^{b}f(x)\mathrm{d}x$ 可以用 MATLAB 程序 `F1 = int(f(x),x,-inf,b)` 计算.

若 $\lim\limits_{a\to-\infty}\int_a^b f(x)\mathrm{d}x$ 存在，则称反常积分 $\int_{-\infty}^{b}f(x)\mathrm{d}x$ 存在或收敛，且 $\int_{-\infty}^{b}f(x)\mathrm{d}x=\lim\limits_{a\to-\infty}\int_a^b f(x)\mathrm{d}x$，否则称反常积分不存在或发散.

(3) 计算 $\int_{-\infty}^{+\infty}f(x)\mathrm{d}x=\int_{-\infty}^{0}f(x)\mathrm{d}x+\int_0^{+\infty}f(x)\mathrm{d}x$ 可以用下面的 MATLAB

程序：

```
>> F1 = int(f(x),x, - inf, + inf)
```

或

```
>> F1 = int(f(x),x, - inf,0)
  F2 = int(f(x),x,0, + inf)
  F = F1 + F2
```

反常积分 $\int_{-\infty}^{+\infty} f(x)\,dx$ 收敛的充要条件是反常积分 $\int_{-\infty}^{0} f(x)\,dx$ 与 $\int_{0}^{+\infty} f(x)\,dx$ 同时收敛，且

$$\int_{-\infty}^{+\infty} f(x)\,dx = \lim_{a\to-\infty}\int_{a}^{0} f(x)\,dx + \lim_{b\to+\infty}\int_{0}^{b} f(x)\,dx.$$

例 9.7.1 讨论反常积分 $I=\int_{1}^{+\infty}\frac{5x^p}{x^4+2}dx$ 的敛散性.

解 输入程序

```
>> syms x
F1 = int((5 * x)/(x^4 +2),x,1, + inf), LimF1 = double(F1)
F2 = int((5 * x.^2)/(x^4 +2),x,1, + inf), LimF2 = double(F2)
F3 = int((5 * x.^3)/(x^4 +2),x,1, + inf), LimF3 = double(F3)
F8 = int((5 * x.^8)/(x^4 +2),x,1, + inf), LimF8 = double(F8)
```

运行后屏幕显示计算 I 的结果 $F_i(i=1,2,3,8)$ 及其近似值 $\lim F_i(i=1,2,3,8)$ 依次如下

```
F1 =
    5/8 * pi * 2^(1/2) -5/4 * atan(1/2 * 2^(1/2)) * 2^(1/2)
LimF1 =
    1.6888
F2 =
    5/4 * 2^(1/4) * pi -5/8 * 2^(1/4) * log(1 -2^(3/4) +2^(1/
2)) +5/8 * 2^(1/4) * log(1 +2^(3/4) +2^(1/2)) -5/4 * 2^(1/4) * atan(2^
(1/4) +1) -5/4 * 2^(1/4) * atan(2^(1/4) -1)
LimF2 =      F3 =      LimF3 =      F8 =      LimF8 =
    3.9734      inf        Inf        inf         Inf
```

由输出的结果可知，当 $p=1,2$ 时，反常积分 $\int_{1}^{+\infty}\frac{5x^p}{x^4+2}dx$ 收敛，且

$$\int_{1}^{+\infty}\frac{5x}{x^4+2}dx\approx 1.688\ 8,\ \int_{1}^{+\infty}\frac{5x^2}{x^4+2}dx\approx 3.973\ 4.$$

当 $p=3,8$ 时，反常积分 $\int_{1}^{+\infty}\frac{5x^p}{x^4+2}dx$ 发散，且

$$\int_{1}^{+\infty}\frac{5x^3}{x^4+2}dx=\infty,\ \int_{1}^{+\infty}\frac{5x^8}{x^4+2}dx=\infty.$$

例 9.7.2 讨论反常积分 $\int_{-\infty}^{-1} \frac{1}{x^{2p}}\mathrm{d}x$ $(p = -2,0.2,0.5,1,4)$的敛散性.

解 当 $p = -2,0.2,0.5,1,4$ 时,输入程序

```
>> syms x
Ff2 = int(1/x^(-2*2),x, -inf,-1), F02 = int(1/x^(2*0.2),x,
-inf,-1)
LimF02 = double(F02), F05 = int(1/x^(0.5*2),x, -inf,-1)
F1 = int(1/x^(1*2),x, -inf,-1), F4 = int(1/x^(4*2),x, -inf,-1)
```

运行后屏幕显示

```
Ff2 =        F02 =                      LimF02 =           F05 =
   Inf          -(-1)^(3/5)*inf    Inf - Infi    -inf
F1 =       F4 =
   1          1/7
```

即当 $p = -2$ 时, $\int_{-\infty}^{-1} \frac{1}{x^{2p}}\mathrm{d}x = \infty$,所以反常积分发散;当 $p = 0.2$ 时, $\int_{-\infty}^{-1} \frac{1}{x^{2p}}\mathrm{d}x = -\infty$,所以反常积分发散;当 $p = 0.5$ 时, $\int_{-\infty}^{-1} \frac{1}{x^{2p}}\mathrm{d}x = -\infty$,所以反常积分发散;当$p = 1$时, $\int_{-\infty}^{-1} \frac{1}{x^{2p}}\mathrm{d}x = 1$,所以反常积分收敛;当 $p = 4$ 时, $\int_{-\infty}^{-1} \frac{1}{x^{2p}}\mathrm{d}x = \frac{1}{7}$,所以反常积分收敛.

例 9.7.3 讨论反常积分 $\int_{-\infty}^{+\infty} \frac{1}{x^2+2x+b}\mathrm{d}x$ $(b = \pm 3)$的敛散性.

解 输入程序如下

```
>>  syms x
  F1 = int(1/(x^2 +2*x +3),x,-inf, +inf), LimF1 = double(F1)
  F2 = int(1/(x^2 +2*x -3),x, -inf,0), LimF2 = double(F2)
  F3 = int(1/(x^2 +2*x -3),x, 0, +inf), LimF3 = double(F3)
  F = F2 + F3,LimF = LimF2 + LimF3
```

运行后屏幕显示计算 $\int_{-\infty}^{+\infty} \frac{1}{x^2+2x+b}\mathrm{d}x$ $(b = \pm 3)$的结果 $F_i(i = 1,2)$及其近似值 $\lim F_i(i = 1,2)$依次如下

```
F1 =                        LimF1 =
   1/2*pi*2^(1/2)        2.221441469079183e+000
F2 =          LimF2 =          F3 =           LimF3 =
  NaN            NaN             NaN              NaN
F =          LimF =
  NaN          NaN
```

即 $\int_{-\infty}^{+\infty}\frac{1}{x^2+2x+3}dx=\frac{\sqrt{2}\pi}{2}$收敛，而 $\int_{-\infty}^{0}\frac{1}{x^2+2x-3}dx$ 不存在，发散. 为什么呢？看了被积函数 $f(x)=\frac{1}{x^2+2x+3}$和 $g(x)=\frac{1}{x^2+2x-3}$的图形（见图 9－5）就明白了.

输入程序

```
>> subplot(1,2,1),
x = -10:0.01:10; y=1./(x.^2 +2 *x+3);
plot(x,y), grid ,title('y=1/(x^2 +2x+3)')
subplot(1,2,2),
x = -10:0.01:10; % z=1./(x.^2 +2 *x-3);
fplot('1./(x.^2 +2 *x-3)',[ -10 10  -2 2]), grid,title('y=1/
(x^2 +2x-3)')
```

运行后屏幕显示图 9－5.

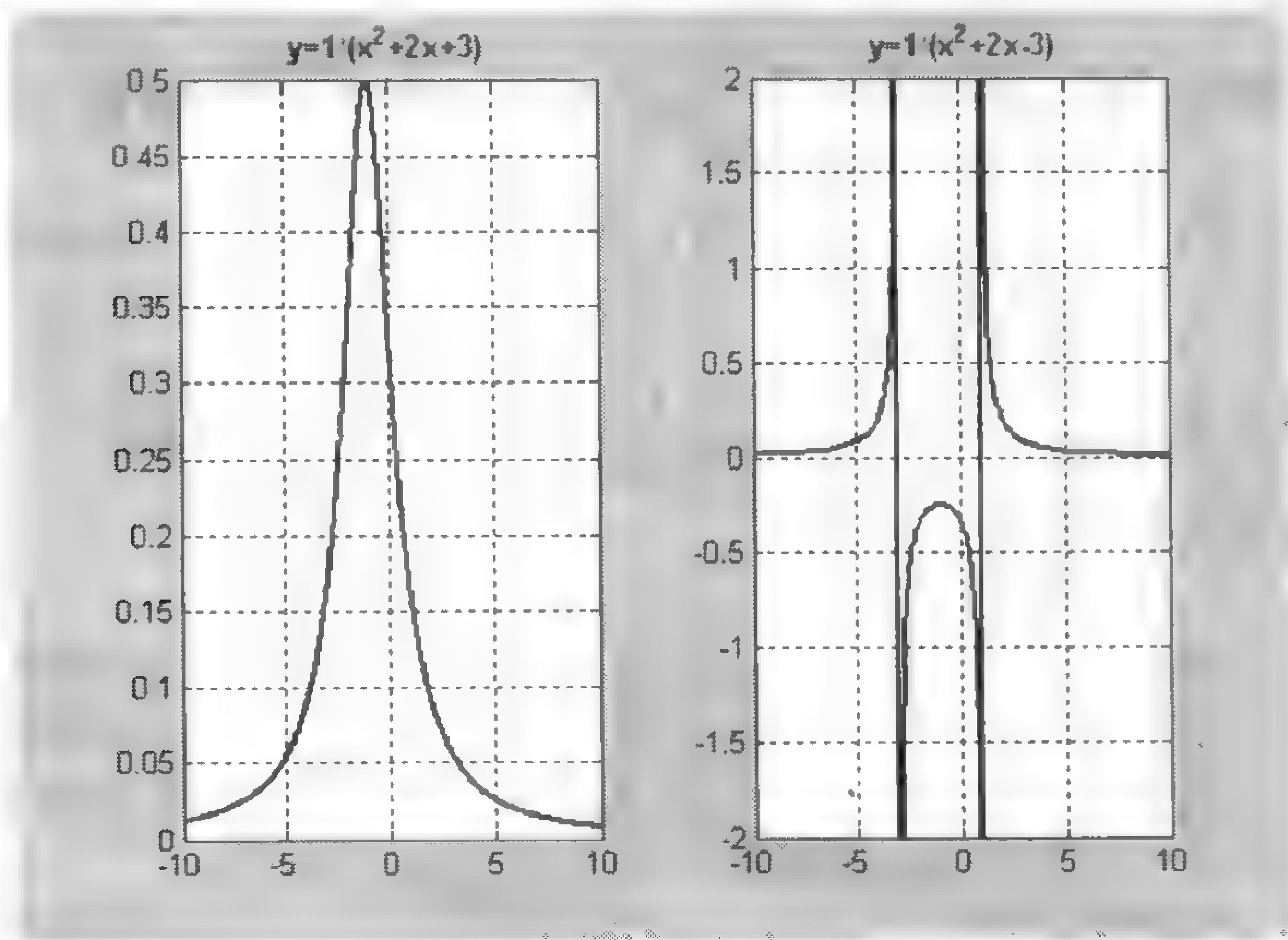


图 9－5 被积函数 $f(x)=\frac{1}{x^2+2x+3}$和 $g(x)=\frac{1}{x^2+2x-3}$的图形

9.7.2 无穷积分的近似计算及其 MATLAB 程序

计算反常积分的近似值的方法很多. 例如，作变量置换法，无穷区间的截断法，累积求和无穷积分法等将反常积分化为有限区间上的有界函数的定积分. 另

外，还可以将本章前面介绍的数值积分的插值型公式，高斯型积分公式等推广到反常积分的近似计算. 下面通过实例加以介绍.

（一）累积求和无穷积分法及其 MATLAB 程序

无穷区间上的反常积分的定义 $\int_a^{+\infty} f(x)\,\mathrm{d}x = \lim\limits_{b\to+\infty}\int_a^b f(x)\,\mathrm{d}x$ 提供了极限过程的原始模式. 设数列 $a=r_0<r_1<r_2<\cdots<r_n<\cdots$ 收敛于 $+\infty$，记

$$\int_a^{+\infty} f(x)\,\mathrm{d}x = \int_a^{r_1} f(x)\,\mathrm{d}x + \int_{r_1}^{r_2} f(x)\,\mathrm{d}x + \cdots, \tag{9.71}$$

其中(9.71)式右端的每项都是正常的定积分，当 $\left|\int_{r_n}^{r_{n+1}} f(x)\,\mathrm{d}x\right| \leqslant \varepsilon$ 时，计算就可以停止. 用这种方法求无穷积分近似值的方法称作**累积求和无穷积分法**. 当然，这只是一种实用性的计算终止条件，而在理论上并不准确. 例如，无穷积分 $\int_1^{+\infty}\frac{1}{x}\mathrm{d}x$ 是发散的，但是用此方法计算结果是一个有限大的数，这与实际结果相矛盾. 为了解决这个问题，采取计算每一步，积分区间放大一倍的方法，即 $r_n=2^n$，加快计算速度. 另外，附加运算所得的值可能小于 ε，则运算过程终止.

下面根据累积求和无穷积分法，由下面公式计算，

$$\int_a^{+\infty} f(x)\,\mathrm{d}x \cong \int_a^{r_1} f(x)\,\mathrm{d}x + \int_{r_1}^{r_2} f(x)\,\mathrm{d}x + \cdots + \int_{r_n}^{r_{n+1}} f(x)\,\mathrm{d}x, \tag{9.72}$$

其中 $r_n=2^n$，当 $\left|\int_{r_n}^{r_{n+1}} f(x)\,\mathrm{d}x\right| \leqslant \varepsilon$ 时，停止计算. 编写的具体 MATLAB 计算程序如下：

> **累积求和无穷积分法计算无穷积分的近似值的 MATLAB 主程序**
>
> 输入量：a 是积分下限，m 是最大累加次数，$wucha$ 是计算终止的条件，即 $\left|\int_{r_n}^{r_{n+1}} f(x)\,\mathrm{d}x\right| \leqslant wucha$.
>
> 输出量：k 是累加次数（即(9.72)式右端的项数），suj 是利用累积求和无穷积分公式(9.72)计算 $\int_a^{+\infty} f(x)\,\mathrm{d}x$ 的近似值，$wugu=\left|\int_{r_n}^{r_{n+1}} f(x)\,\mathrm{d}x\right|$，$F_{jj}=\int_a^{+\infty} f(x)\,\mathrm{d}x$，$WC=|F_{jj}-suj|$.

名为 wqjfjx.m 的累积求和无穷积分法的 MATLAB 主程序如下：

```
function [k,suj,wugu,Fjj,WC] =wqjfjx(a,Wucha,m)
r =a;k =0; wugu =1;
syms t
Y =input('计算没有结束,请输入被积函数 f(x) =');% exp( -t)./(1 +t.^4);
su =int(Y,t,a,a +2)
suj = double(su)
```

```
while ((wugu >Wucha)&(k <m))
  k =k +1,r =a +2^k,r1 =a +2^(k +1)
  syms t
  Y =input('计算没有结束,请输入被积函数 f(x) =');
  F1 =int(Y,t,r,r1), intF1 = double(F1);
  suj =suj +intF1,wugu =abs(intF1)
  Fj -int(Y,t,0, +inf)
  Fjj = double(Fj), WC =abs(Fjj - suj)
end
disp('k 是累加次数,suj 是利用累积求和无穷积分法计算 f(x)从 a 到正无穷
的无穷积分的近似值,wugu 是 f(x)在[r(n),r(n +1)]上的积分的绝对值,Fjj 是 f(x)
从 a 到正无穷的无穷积分的值,WC = |Fjj - suj|')
```

说明:用此程序时,需要用户根据计算机屏幕的提示在 MATLAB 工作窗口输入被积函数(例如,exp(- t)./(1 + t.^4)),输入以后,回车才继续运行.如果计算结果不满足给定的计算终止的条件,计算机屏幕会给出提示:'计算没有结束,请输入被积函数 f(x) =';输入被积函数以后,回车才继续运行,…….如此继续下去,当 $wugu<10^{-6}$或累加次数大于给定的最大累加次数时,计算机停止运算,屏幕显示输出量的说明和计算结果.

实际上,有时计算的近似值 suj 与精确值 F_{jj}的截断误差小于要求的精度 ε.但是,有时计算的近似值 suj 与精确值 F_{jj}的截断误差也可能大于要求的精度 ε.如果对于计算结果的精度的要求非常高,这是不行的.解决这个问题的常用方法有两种,一种方法是给定的 $wucha$ 大于实际要求的精度;另外一种方法是先求出(9.72)式的项数 k,然后再计算,这样近似值与精确值就达到要求的精度.

例 9.7.4 计算 $I=\int_0^{+\infty}\frac{1}{e^x(1+x^4)}dx$,使精确度为 10^{-6},并与精确值比较.

解 取 $r_n=2^n$,根据(9.72)式计算 $I=\sum_{k=0}^{+\infty}\int_{r_k}^{r_{k+1}}\frac{1}{e^x(1+x^4)}dx$.输入下面的程序

```
>> a =0; Wucha  =1.e -6;m =100;
[k,suj,wugu,Fjj,WC] =wqjfjx(a,Wucha,m)
```

运行后屏幕显示

```
计算没有结束,请输入被积函数 f(x) =
```

输入函数

```
exp( -t)./(1 +t.^4);
```

运行后屏幕显示

```
Warning: Explicit integral could not be found.
```

```
> In C:\MATLAB6p5\toolbox\symbolic\@ sym\int.m at line 58
  In C:\MATLAB6p5\work\wqjfjx.m at line 5
su =
int(exp( -t)/(1 +t^4),t = 0 ..2)
suj =
    0.62745967782312
k =
      1
r =
      2
r1 =
      4
```

计算没有结束,请输入被积函数 f(x) =

输入函数

```
exp( -t)./(1 +t.^4);
```

运行后屏幕显示提示,……,如此继续下去,当 $wugu<10^{-6}$ 时,屏幕显示如下内容

k 是累加次数,suj 是利用累积求和无穷积分法计算 f(x)从 a 到正无穷的无穷积分的近似值,wugu 是 f(x)在[r(n),r(n+1)]上的积分的绝对值,Fjj 是 f(x)从 a 到正无穷的无穷积分的值,WC = |Fjj - suj|

```
k =
      3
suj =
    0.63047783491711
wugu =
    5.598881094648278e-008
Fjj =
    0.63047783491850
WC =
    1.386668557756821e-012
```

取前三项的和作为近似值,即 $I\approx\sum\limits_{k=0}^{3}\int_{r_k}^{r_{k+1}}\frac{1}{e^x(1+x^4)}dx=F_{jj}\approx 0.630\ 478.$ 用计算机计算,F_{jj} 与精确值 0.630 477 834 918 50 … 的绝对误差约为 1.386 669e - 012,$\left|\int_{r_3}^{r_4}f(x)\,dx\right|\approx 5.598\ 881e-008$,都达到预先给定的精度 10^{-6}.

(二) 无穷区间的截断法

对无穷区间的积分 $\int_a^{\infty}f(x)\,dx$,可以将被积函数 $f(x)$ 的"尾巴"截去,化为有

限区间的积分,正确运用此法的关键是要事先用某种简单的解析方法估计出截断部分的数值,使之满足精度要求.

例 9.7.5 用数值方法计算 $I=\int_0^{+\infty} e^{-3x^2}dx$,使精确度为 10^{-12},并与精确值比较.

解 为估计截断部分 $\int_N^{+\infty} e^{-3x^2}dx$ 的值,利用 $x^2 \geqslant Nx\ (x \geqslant N)$ 时

$$\int_N^{+\infty} e^{-3x^2}dx \leqslant \int_N^{+\infty} e^{-3Nx}dx = \frac{1}{3N}e^{-3N^2} \leqslant 10^{-12}.$$

将下面的函数保存为名为 nm.m 的 M 文件

```
function y = nm(x)
y = exp( -3 * x.^2)./(3 * x);
```

输入程序

```
>> N = 2:5;y = nm(N)
```

运行后屏幕显示如下

```
y =   1.0e-005 *
0.10240353922214  0.00000002088365  0.00000000000000  0.00000000000000
```

由此可见,如果不计算误差传播,取 $N=3$,可满足 $\left|I-\int_0^3 e^{-x^2}dx\right| \leqslant 10^{-12}$.

输入程序

```
>> x = 0:0.1:3; y = exp( -3 * x.^2); z = trapz(x,y)
syms t, F1 = int(exp( -3 * t.^2),t,0, +inf),
F3 = int(exp( -3 * t.^2),t,0,3)
intF1 = double(F1), Wuchaz = abs(intF1 - z)
intF3 = double(F3), Wucha3 = abs(intF1 - intF3)
```

运行后屏幕显示当 $N=3$ 时,分别用梯形公式和 int 函数计算 $\int_0^3 e^{-x^2}dx$ 的近似值 z 和精确解 F_3(或 $intF_3$)及其与 I 的误差 *Wuchaz* 和 *Wucha3* 如下

```
z =                     F1 =                       intF1 =
  0.51166335397311       1/6*3^(1/2)*pi^(1/2)      0.51166335397324
F3 =                                               intF3 =
  1/6*erf(3*3^(1/2))*3^(1/2)*pi^(1/2)              0.51166335397314
Wuchaz =                                  Wucha3 =
  1.294520046712933e-013                  1.025846074753645e-013
```

由输出结果可见,z 和 $intF_3$ 分与 I 的误差 *Wuchaz* 和 *Wucha3* 都小于 10^{-12},故

$$I=\frac{\sqrt{3\pi}}{6} \approx 0.511\ 663\ 353\ 973,$$

例 9.7.6 如果取 $r_k=2k\pi,k=0,1,2,\cdots$，根据(9.72)式，当 k 取何值时，$I=\int_0^{+\infty}\frac{\sin x}{1+x^2}\mathrm{d}x$ 的近似值的截断误差为 10^{-6}，计算其近似值，并与精确值比较.

解 (1) 首先求 k 的值.

取 $r_k=2k\pi,k=0,1,2,\cdots$，根据(9.72)式计算

$$\left|\int_{2k\pi}^{+\infty}\frac{\sin x}{1+x^2}\mathrm{d}x\right|=\left|\sum_{k=n}^{+\infty}\int_{2k\pi}^{2(k+1)\pi}\frac{\sin x}{1+x^2}\mathrm{d}x\right|<\frac{1}{4k^2\pi}\leqslant 10^{-6}.$$

输入下面的程序

```
>> k = ceil(sqrt(10^6 /(4 * pi)))
```

运行后屏幕显示

```
k =
    283
```

因此，当 $k\geqslant 283$ 时，I 的近似值的截断误差为 10^{-6}.

(2) 然后计算 I 的近似值，使截断误差为 10^{-6}，并与精确值比较.

编写并保存名为 wqjfjx1.m 的计算主程序如下

```
function [k,suj,wugu] = wqjfjx1(a,m)
r = a;k = 0; wugu = 1; syms t
fun = sin(t)./(1 + t.^2); su = int(fun,t,a,2 * pi); suj = double(su);
while (k < m)
k = k + 1,r = 2 * k * pi; r1 = 2 * (k + 1) * pi;
syms t
F1 = int(fun,t,r,r1); intF1 = double(F1); suj = suj + intF1, wugu
 = abs(intF1),
end
```

输入下面的程序

```
>>  a = 0,m = 283,[k,suj,wugu] = wqjfjx1(a,m)
syms t
fun = sin(t)./(1 + t.^2); Fj = int(fun,t,0, + inf), Fjj = double(Fj)
WC = abs(Fjj - suj)
```

运行后屏幕显示计算过程和结果如下：

```
k =   suj =                                  wugu =
 1     0.64068026831490 - 0.00000000000000i  0.01602703309687
k =   suj =                                  wugu =
 2     0.64399926393154 - 0.00000000000000i  0.00331899561664
k =   suj =                                  wugu =
 3     0.64519499777913 - 0.00000000000000i  0.00119573384758
```

```
........................................
k =    suj =                                      wugu =
283  0.64676080872636 - 0.00000000000000i  2.223370580497967e-009
Fj =
 -i*sinint(i)*cosh(1)-cosint(i)*sinh(1)+1/2*i*sinh(1)*pi
Fjj =                                        WC =
  0.64676112277913 - 0.00000000000000i      3.140527699008189e-007
```

由上面的输出结果可以看出,当 $k=283$ 时,计算机计算出的 I 的近似值 $suj=0.646\,760\,808\,726\,36$ 与精确值 $F_{jj}=0.646\,761\,122\,779\,13\cdots$ 的截断误差约为 $3.140\,527\,699\,008\,19\text{e}-007$ 小于要求的精度 10^{-6}. 故

$$I\approx\sum_{k=0}^{283}\int_{2k\pi}^{2(k+1)\pi}\frac{\sin x}{1+x^2}\mathrm{d}x\approx 0.646\,761,$$

满足

$$\left|\int_0^{+\infty}\frac{\sin x}{1+x^2}\mathrm{d}x-\int_0^{568\pi}\frac{\sin x}{1+x^2}\mathrm{d}x\right|=\left|\int_{568\pi}^{+\infty}\frac{\sin x}{1+x^2}\mathrm{d}x\right|<10^{-6},$$

且

$$\left|\int_{2k\pi}^{2(k+1)\pi}\frac{\sin x}{1+x^2}\mathrm{d}x\right|\approx 2.223\,371\text{e}-009<10^{-6}.$$

如果对于计算结果的精度的要求非常高,求出(9.72)式的项数 k,使近似值与精确值达到要求的精度是十分必要的.

(三) 高斯-拉盖尔(Gauss-Laguerre)求积公式及其 MATLAB 程序

在无穷区间 $[0,+\infty)$ 上的 n 次正交多项式——拉盖尔多项式

$$U_n(x)=\mathrm{e}^x\frac{d^n}{dx^n}(x^n\mathrm{e}^{-x}),\ 0\leqslant x<+\infty \tag{9.73}$$

满足递推关系式

$$U_{n+1}(x)=(1+2n-x)U_n(x)-n^2U_{n-1}(x),n=1,2,\cdots,$$
$$U_0(x)=1,$$
$$U_1(x)=1-x.$$

计算可得

$$U_0(x)=1,$$
$$U_1(x)=1-x,$$
$$U_2(x)=x^2-4x+2,$$
$$U_3(x)=-x^3+9x^2-18x+6,$$
$$U_4(x)=x^4-16x^3+72x^2-96x+24,$$
$$U_5(x)=-x^5+25x^4-200x^3+600x^2-600x+120,$$
$$U_6(x)=x^6-36x^5+450x^4-2\,400x^3+5\,400x^2-4\,320x+720,$$

$U_7(x) = -x^7 + 49x^6 - 882x^5 + 7\,350x^4 - 29\,400x^3 + 52\,920x^2 - 35\,280x + 5\,040$,

取 n 次正交多项式——拉盖尔多项式(9.73)的零点 x_k 作求积节点,则形成的函数 $f(x)$ 在 $[0,+\infty)$ 上的 n 个节点的**高斯 - 拉盖尔无穷数值积分公式**

$$G_n(f) = A_{n,1}f(x_1) + A_{n,2}f(x_2) + \cdots + A_{n,n}f(x_n), \tag{9.74}$$

其中

$$A_k = \int_0^{+\infty} \frac{e^{-x}U_n(x)}{(x-x_k)U'_n(x_k)}dx = \frac{(n!)^2}{x_k[U'_n(x_k)]^2},\ k=1,2,\cdots,n, \tag{9.75}$$

使得

$$\int_0^{+\infty} e^{-x}f(x)\,dx = G_n(f) + R_n[f,G_n],$$

其中截断误差为

$$R_n[f,G_n] = \frac{f^{(2n)}(\xi)}{a_n^2(2n)!}\int_0^{+\infty} e^{-x}U_n^2(x)\,dx,$$

得

$$R_n[f,G_n] = \frac{(n!)^2}{(2n)!}f^{(2n)}(\xi)\ \ (0<\xi<+\infty).$$

将所需要的常用的函数 $f(x)$ 在 $[0,+\infty)$ 上的 n 个节点的高斯 - 拉盖尔无穷数值积分公式(9.74)中的节点的横坐标 x_k 和系数 $A_{n,k}$ $(k=1,2,\cdots,6)$ 已经制成表 9-7,用时直接查找即可.

表 9-7 常用的高斯 - 拉盖尔无穷数值积分公式表

$\int_0^{+\infty} e^{-x}f(x)\,dx \approx \sum_{k=1}^{n} A_{n,k}f(x_k)$					
n	横坐标 x_k	系数 $A_{n,k}$	n	横坐标 x_k	系数 $A_{n,k}$
1	1	1	5	0.263 560 319 718 14	0.521 755 610 582 81
2	3.414 213 562 373 09	0.036 611 652 351 68		1.413 403 059 106 52	0.398 666 811 08 317
	0.585 786 437 626 90	0.213 388 347 648 32		3.596 425 771 040 72	0.075 942 449 681 71
3	6.289 945 082 937 48	0.010 389 256 501 59		7.085 810 005 858 84	0.003 611 758 679 92
	0.415 774 556 783 48	0.7110 930 099 29 17		12.640 800 844 275 78	0.000 023 369 972 39
	2.294 280 360 279 04	0.278 517 733 569 24			
4	9.395 070 912 301 13	0.000 033 705 919 10	6	0.222 846 604 179 26	0.012 749 018 720 83
	4.536 620 296 921 13	0.002 430 494 282 19		1.188 932 101 672 62	0.011 583 356 410 34
	1.745 761 101 158 35	0.022 338 668 277 36		2.992 736 326 059 31	0.003 149 260 613 17
	0.322 547 689 619 39	0.037 697 131 521 35		5.775 143 569 104 51	0.000 288 866 595 92
				9.837 467 418 382 59	0.000 007 250 477 86
				15.982 873 980 601 70	0.000 000 024 959 66

根据(9.73)式、(9.74)式和(9.75)式编写的求 n 次正交多项式——拉盖尔多项式、节点的横坐标 x_k 和系数公式 $A_{n,k}(k=1,2,\cdots,n)$ 及其在 x_k 处的值的 MATLAB 程序，读者也可以用这些程序计算.

计算 n 次正交多项式——拉盖尔多项式 $U_n(x)$ 和系数公式 $A_{n,k}(k=1,2,\cdots,n)$ 的 MATLAB 主程序

输入量：n 是正交多项式——拉盖尔多项式 $U_n(x)$ 的次数.

输出量：UK 是 0 至 n 次正交多项式——拉盖尔多项式 $U_n(x)$，AK 是系数公式 $A_{n,k}(k=1,2,\cdots,n)$，K 是每个多项式 UK 的次数和系数公式 AK 的角标.

```
function [K,UK,AK] = laguerre(n)
syms x
K = 0,U0 = 1,K = 1,U1 = 1 - x,u1 = 1;u2 = 1 - x;
duk = diff(u2,x); n = n - 1;sun1 = 1;
duk2 = duk^2;
A1 = (sun1^2)/(x * duk2);A1 = simple(A1),
sun = 1;
for k = 1:2:n
   u3 = (1 + 2 * k - x) * u2 - (k^2) * u1;K = k + 1, UK = simple(u3),
   duk1 = diff(u3,x);
   sun2 = sun1 * K;
   duk21 = duk1^2;
   Ak1 = (sun1^2)/(x * duk21); AK = simple(Ak1),
   u4 = (1 + 2 * (k + 1) - x) * u3 - ((k + 1)^2) * u2;
   K = k + 2, UK = simple(u4),
   duk = diff(u4,x);
   sun3 = sun2 * K;sun1 = sun3;
   duk2 = duk^2;
   Ak2 = (sun3^2)/(x * duk2); AK = simple(Ak2),
   u1 = u3;u2 = u4;
 if k > n,break,end;
 end
 K = k + 2; UK; AK;
```

例 9.7.7 根据(9.73)式、(9.74)式和(9.75)式，计算 1 至 7 次正交多项式——拉盖尔多项式、节点的横坐标 x_j $(j=1,2,3,4,5,6,7)$ 和系数 $A_{n,k}$ $(k=1,2,\cdots,7)$ 公式及其在 $x_j(j=1,2,3,4,5,6,7)$ 处的值 A_k.

解 (1) 建立并保存以 laguerre.m 文件命名的 M 文件函数.

(2) 计算 1 至 7 次正交多项式——拉盖尔多项式和系数 $A_{n,k}(k=1,2,\cdots,7)$公式. 输入程序

```
>>n=7,[K,U,Ak]=laguerre(n)
```

运行后输出结果

```
K =
      0
U0 =
      1
K =
      1
U1 =
   1 - x
A1 =
   1/x
K =
      2
UK =
   2 - 4 * x + x^2
AK =
   1/x/( - 4 + 2 * x)^2
K =
      3
UK =
  6 - 18 * x + 9 * x^2 - x^3
AK =
  4/x/(6 - 6 * x + x^2)^2
K =
      4
UK =
  - 16 * x^3 + 72 * x^2 - 96 * x + 24 + x^4
AK =
  9/4/x/( - 24 + 36 * x - 12 * x^2 + x^3)^2
K =
      5
UK =
   - 200 * x^3 + 600 * x^2 - 600 * x + 120 + 25 * x^4 - x^5
AK =
   576/x/(120 * x^2 - 240 * x + 120 - 20 * x^3 + x^4)^2
```

```
K =
     6
UK =
  720 -4320 * x +5400 * x^2 -2400 * x^3 +450 * x^4 -36 * x^5 +x^6
AK =
  400/x/( -720 +1800 * x -1200 * x^2 +300 * x^3 -30 * x^4 +x^5)^2
K =
     7
UK =
  5040 -35280 * x +52920 * x^2 -29400 * x^3 +7350 * x^4 -882 * x^5
   +49 * x^6 -x^7
AK =
  518400/x/( -15120 * x +5040 -4200 * x^3 +12600 * x^2 -42 * x^5
   +630 * x^4 +x^6)^2
```

(3) 求 1 至 7 次正交多项式——拉盖尔多项式的零点 x_k，即求积节点. 输入程序

```
>> syms x y
y1 =('1 -x =0')
y2 =('x^2 -4 * x +2 =0')
y3 =('-x^3 +9 * x^2 -18 * x +6 =0')
y4 =('x^4 -16 * x^3 +72 * x^2 -96 * x +24 =0')
y5 =('-x^5 +25 * x^4 -200 * x^3 +600 * x^2 -600 * x +120 =0')
y6 =('720 -4320 * x +5400 * x^2 -2400 * x^3 +450 * x^4 -36 * x^5 +x^
6 =0')
y7 =('5040 -35280 * x +52920 * x^2 -29400 * x^3 +7350 * x^4 -882 *
x^5 +49 * x^6 -x^7 =0')
y11 =solve(y1,x), x1 =double(y11)
y22 =solve(y2,x) , x2 =double(y22),
y33 =solve(y3,x) , x3 =double(y33),
y44 =solve(y4,x) , x4 =double(y44),
y55 =solve(y5,x) , x5 =double(y55),
y66 =solve(y6,x) , x6 =double(y66),
y77 =solve(y7,x) , x7 =double(y77),
```

运行后可以得到(9.74)式中的节点的横坐标 $x_k(k=1,2,3,4,5,6,7)$(略).

(4) 求在节点的横坐标 $x_j(j=1,2,3,4,5,6,7)$处的系数 $A_k(k=1,2,\cdots,7)$. 输入程序

```
>> x1 =[1],A1 =1./x1
x2 =[3.41421356237309,0.58578643762690],A2 =1./x2./( -4 +2 *
```

```
x2).^2
        x3 =[6.28994508293748  0.41577455678348  2.29428036027904],
        A3 =4./x3./(6 -6 * x3 +x3.^2).^2
        x4 =[9.39507091230113  4.53662029692113  1.74576110115835
0.32254768961939],
        A4  =9/4./x4./( -24 +36 * x4 -12 * x4.^2 +x4.^3).^2
        x5 =[0.26356031971814  1.41340305910652  3.59642577104072
7.08581000585884  12.64080084427578],
        A5  =576./x5./(120 -240 * x5 +120 * x5.^2 -20 * x5.^3 +x5.^4).^2
        x6 =[0.22284660417926  1.18893210167262  2.99273632605931
5.77514356910451  9.83746741838259  15.98287398060170],
        A6  =400./x6./( -720 +1800 * x6 -1200 * x6.^2 +300 * x6.^3 -30 *
x6.^4 +x6.^5).^2
        x7 =[0.19304367656036  1.02666489533919  2.56787674495075
4.90035308452648  8.18215344456286  12.73418029179782  19.39572786226254],
        A7  =518400./x7./(5040 -15120 * x7 +12600 * x7.^2 -4200 * x7.^3
 +630 * x7.^4 -42 * x7.^5 +x7.^6).^2
```

运行后输出节点的横坐标 $x_j(j=1,2,3,4,5,6,7)$ 和对应的系数 $A_k(k=1,2,\cdots,7)$ 如下

```
        x1 =
              1
        A1 =
              1
        x2 =
           3.41421356237309   0.58578643762690
        A2 =
           0.03661165235168   0.21338834764832
        x3 =
           6.28994508293748   0.41577455678348   2.29428036027904
        A3 =
           0.01038925650159   0.71109300992917   0.27851773356924
        x4 =
           9.39507091230113   4.53662029692113   1.74576110115835
0.32254768961939
        A4 =
          0.00003370591910   0.00243049428219   0.02233866827736
0.03769713152135
        x5 =
```

```
   Columns 1 through 4
  0.26356031971814   1.41340305910652   3.59642577104072
7.08581000585884
   Column 5
  12.64080084427578
 A5 =
   Columns 1 through 4
  0.52175561058281   0.39866681108317   0.07594244968171
0.00361175867992
   Column 5
  0.00002336997239
 x6 =
   Columns 1 through 4
  0.22284660417926   1.18893210167262   2.99273632605931
5.77514356910451
   Columns 5 through 6
  9.83746741838259  15.98287398060170
 A6 =
   Columns 1 through 4
  0.01274901872083   0.01158335641034   0.00314926061317
0.00028886659592
   Columns 5 through 6
  0.00000725047786   0.00000002495966
 x7 =
   Columns 1 through 4
  0.19304367656036   1.02666489533919   2.56787674495075
4.90035308452648
   Columns 5 through 7
  8.18215344456286  12.73418029179782  19.39572786226254
 A7 =
   Columns 1 through 4
   0.40931895170127   0.42183127786172   0.14712634865750
0.02063351446872
   Columns 5 through 7
   0.00107401014328   0.00001586546435   0.00000003170315
```

根据函数 $f(x)$ 在 $[0,+\infty)$ 上的 n 个节点的高斯－拉盖尔无穷数值积分公式编写的计算 $\int_0^{+\infty} e^{-x}f(x)\,dx$ 的数值积分及其截断误差公式的 MATLAB 程序

如下：

用高斯－拉盖尔无穷积分公式计算 $\int_0^{+\infty} e^{-x}f(x)dx$ 的数值积分及其截断误差公式的 MATLAB 主程序

输入量：*fun* 是被积函数中的因子 $f(x)$，X 是 n 个节点的横坐标 x_k 的 $1\times n$ 维向量，A 是系数 $A_{n,k}(k=1,2,\cdots,n)$ 的 $1\times n$ 维向量，其中 X 和 A 可以从表 9－7中查取或用上述的 MATLAB 程序求得.

输出量：*GL* 是利用 n 个节点的高斯－拉盖尔无穷积分公式计算 $\int_0^{+\infty} e^{-x}f(x)dx$ 的数值积分值，*RGn* 是截断误差公式，其中 M 是 $f(x)$ 的 $2n$ 阶导数，Y 是 $f(x)$ 在 X 处的值.

```
function [GL,Y, RGn] =GaussL1(fun,X,A)
n = length(X);n2 =2 * n; Y = feval(fun,X);
GL = sum(A.* Y); sun =1; su2n =1; su2n1 =1;
for k =1:n
  sun = sun * k;
end
for k =1:n2
  su2n = su2n * k;
end
syms M
RGn = ( sun^2) * M/( su2n);
```

另外，根据高斯－拉盖尔无穷积分公式编写的用 n 个节点的高斯－拉盖尔无穷积分公式计算 $\int_0^{+\infty} e^{-x}f(x)dx$ 的数值积分及其估计绝对误差限的 MATLAB 程序如下：

用高斯－拉盖尔无穷积分公式计算 $\int_0^{+\infty} e^{-x}f(x)dx$ 的数值积分和误差估计的 MATLAB 主程序

输入量：*fun* 是被积函数中的因子 $f(x)$，*fun2n* 是 $f(x)$ 的 $2n$ 阶导数，X 是 n 个节点的横坐标 x_k 的 $1\times n$ 维向量，A 是系数 $A_{n,k}(k=1,2,\cdots,n)$ 的 $1\times n$ 维向量，其中 X 和 A 可以从表 9－7 中查取或用上述的 MATLAB 程序求得.

输出量：*Rn* 是高斯－拉盖尔无穷积分公式的误差估计，*GL* 是高斯－拉盖尔无穷积分公式计算 $\int_0^{+\infty} e^{-x}f(x)dx$ 的数值积分值.

```
function [GL,Y,Rn] = GaussL2 (fun,X,A,fun2n)
n = length(X);n2 = 2 * n; Y = feval(fun,X);
GL = sum(A.* Y); sun = 1; su2n = 1; su2n1 = 1;
for k = 1:n
  sun = sun * k;
end
for k = 1:n2
  su2n = su2n * k;
end
mfun2n1  = max(fun2n);
mfun2n = abs(mfun2n1);
Rn = ( sun^2) * mfun2n /( su2n);
```

例 9.7.8 用高斯－拉盖尔无穷积分公式计算 $\int_0^{+\infty}\frac{1}{e^x(5+x^2)}dx$，取 $n=7$，再根据截断误差公式写出误差公式，并将计算结果与精确值进行比较.

解 (1) 建立并保存以 fun.m 文件命名的 M 文件函数

```
function y = fun(x)
        y = 1./(5 + x.^2);
```

(2) 建立并保存用 GaussL2.m 文件命名的 M 文件函数.

(3) 输入程序

```
>> syms x
fun2n = diff(exp( - x)./(5 + x^2),x,10); fun2ns = simple(fun2n)
```

(4) 运行后,再输入程序

```
>> X = [0.19304367656036,1.02666489533919,2.56787674495075,
4.90035308452648, 8.18215344456286,12.73418029179782,19.39572786226254];
A - [0.40931895170127,0.42183127786172,0.14712634865750,
0.02063351446872,0.00107401014328,0.00001586546435,0.00000003170315];
x = 0:0.001:1000;
fun2ns = exp( - x). * (50889062500 * x - 29608750000 * x.^2 +
89352450000 * x.^5 + 69284490000 * x.^6 + 9796290000 * x.^7 - 185554687500 *
x.^3 - 70161046875 * x.^4 - 6677396250 * x.^8 - 868165200 * x.^10 -
3734325000 * x.^9 + 14759850 * x.^12 + 1748400 * x.^14 - 79569000 * x.^11 +
7568400 * x.^13 + 36675 * x.^16 + 320 * x.^18 + x.^20 + 285840 * x.^15 + 3780 *
x.^17 + 20 * x.^19 + 7000234375)./(5 + x.^2).^11;
[GL,Y, Rn] = GaussL2(@ fun,X,A,fun2ns)
syms t
fi = int(exp( - t)/(5 + t^2),t,0, + inf);
Fs = double (fi), wGL = double (abs(fi - GL))
```

运行后屏幕显示取 $n=7$,用高斯-拉盖尔无穷积分公式计算 I 的结果 GL 和精确值 F_s 及其两者的绝对误差 wGL 依次如下

```
GL =
    0.16435187219536
Y =
  Columns 1 through 4
0.19852039332099  0.16517893285268  0.08625157652455  0.03446676087003
  Columns 5 through 7
  0.01389899751582  0.00598231575880  0.00262333436453
Rn =
    0.06539672874780
Fs =
    0.16435041671009 + 0.00000000000000i
wGL =
    1.455485269162850e-006
```

(四)高斯-埃尔米特(Gauss-Hermite)求积公式及其 MATLAB 程序

如果在无穷区间$(-\infty,+\infty)$上,取 n 次正交多项式——埃尔米特多项式为

$$H_n(x)=(-1)^n e^{x^2}\frac{d^n}{dx^n}(e^{-x^2}) \tag{9.76}$$

的零点 x_k 作求积节点,则函数 $f(x)$ 在$(-\infty,+\infty)$上的 n 个节点的高斯-埃尔米特无穷数值积分公式为

$$H_n(f)=A_{n,1}f(x_1)+A_{n,2}f(x_2)+\cdots+A_{n,n}f(x_n), \tag{9.77}$$

其中

$$A_k=\int_{-\infty}^{+\infty}\frac{e^{-x^2}H_n(x)}{(x-x_k)H'_n(x_k)}dx=\frac{2^{n+1}\sqrt{\pi}n!}{[H'_n(x_k)]^2},\ k=1,2,\cdots,n,$$

使得

$$\int_{-\infty}^{+\infty}e^{-x^2}f(x)dx=H_n(f)+R_n[f,H_n]. \tag{9.78}$$

因为(9.76)式的最高次数项的系数 $a_n=2^n$,所以**高斯-埃尔米特无穷数值积分公式(9.77)的截断误差为**

$$R_n[f,H_n]=\frac{f^{(2n)}(\xi)}{a_n^2(2n)!}\int_{-\infty}^{+\infty}e^{-x^2}H_n^2(x)dx\ (-\infty<\xi<+\infty).$$

即

$$R_n[f,H_n]=\frac{n!\sqrt{\pi}}{2^n(2n)!}f^{(2n)}(\xi)\ (-\infty<\xi<+\infty). \tag{9.79}$$

将所需要的常用的函数 $f(x)$ 在$(-\infty,+\infty)$上的 n 个节点的高斯-埃尔米

特无穷数值积分公式中的节点的横坐标 x_k 和系数 $A_{n,k}(k=1,2,\cdots,n)$ 已经制成表 9-8,用时直接查找即可.

表 9-8 常用的高斯-埃尔米特无穷数值积分公式表

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)\,dx \approx \sum_{k=1}^{n} A_{n,k} f(x_k)$$

n	横坐标 x_k	系数 $A_{n,k}$	n	横坐标 x_k	系数 $A_{n,k}$
1	0	1.772 453 850 0	6	2.350 604 973 7	0.004 530 009 9
2	0.707 106 781 2	0.886 226 925 5		-2.350 604 973 7	0.004 530 009 9
	-0.707 106 781 2	0.886 226 925 5		1.335 849 074 0	0.157 067 320 3
3	1.224 744 871 4	0.295 408 975 2		-1.335 849 074 0	0.157 067 320 3
	-1.224 744 871 4	0.295 408 975 2		0.436 077 411 9	0.724 629 595 2
	0	1.181 635 900 6		-0.436 077 411 9	0.724 629 595 2
4	1.650 680 123 9	0.081 312 835 5	7	2.651 961 356 8	0.000 971 781 2
	-1.650 680 123 9	0.081 312 835 5		-2.651 961 356 8	0.000 971 781 2
	0.524 647 623 3	0.804 914 090 0		1.673 551 628 8	0.054 515 582 8
	-0.524 647 623 3	0.804 914 090 0		-1.673 551 628 8	0.054 515 582 8
				0.816 287 882 9	0.425 605 252 6
				-0.816 287 882 9	0.425 605 252 6
				0	0.810 264 617 6
5	2.020 182 870 5	0.019 953 242 1	8	2.930 637 420 3	0.000 199 604 1
	-2.020 182 870 5	0.019 953 242 1		-2.930 637 420 3	0.000 199 604 1
	0.958 572 464 6	0.393 619 323 2		1.981 656 756 7	0.017 077 983 0
	-0.958 572 464 6	0.393 619 323 2		-1.981 656 756 7	0.017 077 983 0
	0	0.945 308 720 5		1.157 193 712 4	0.207 802 325 8
				-1.157 193 712 4	0.207 802 325 8
				0.381 186 990 2	0.661 147 012 6
				-0.381 186 990 2	0.661 147 012 6

根据函数 $f(x)$ 在 $(-\infty,+\infty)$ 上的 n 个节点的高斯-埃尔米特无穷数值积分公式,编写的计算 $I=\int_{-\infty}^{+\infty} e^{-x^2} f(x)\,dx$ 的数值积分及其截断误差公式的 MATLAB 程序如下:

高斯－埃尔米特无穷数值积分公式计算 $\int_{-\infty}^{+\infty} e^{-x^2} f(x)\,dx$ 及其截断误差公式的 MATLAB 主程序

输入量:*fun* 是被积函数中的因子 $f(x)$,X 是 n 个节点的横坐标 x_k 的 $1\times n$ 维向量,A 是系数 $A_{n,k}(k=1,2,\cdots,n)$ 的 $1\times n$ 维向量,其中 X 和 A 可以从表 9－8中查取或自编 MATLAB 程序求得.

输出量:*GH* 是利用 n 个节点的高斯－埃尔米特无穷积分公式计算 I 的数值积分值,*RHn* 是截断误差公式,其中 M 是 $f(x)$ 的 $2n$ 阶导数,Y 是 $f(x)$ 在 X 处的值.

```
function [GH,Y,RHn] = GaussH1(fun,X,A)
n = length(X);n2 = 2 * n; Y = feval(fun,X); GH = sum(A.* Y);
sun = 1; su2n = 1; su2n1 = 1;
for k = 1:n
  sun = sun * k;
end
for k = 1:n2
  su2n = su2n * k;
end
syms M
RHn = ( sun * sqrt(pi)) * M/( 2^n * su2n);
```

例 9.7.9 用高斯－埃尔米特无穷积分公式计算 $I=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} e^{-x^2}\sin 3x\,dx$,取 $n=5$,再根据截断误差公式写出误差公式,并将计算结果与精确值进行比较.

解 (1) 建立并保存 fun.m 文件命名的 M 文件函数

```
function y = fun (x)
      y = sin(3 * x)./sqrt(2 * pi);
```

(2) 建立并保存 GaussH1.m 文件命名的 M 文件函数.

(3) 根据表 9－8 ,输入程序

```
>> X = [2.0201828705 -2.0201828705 0.9585724646 -0.9585724646 0];
A = [0.0199532421  0.0199532421  0.3936193232  0.3936193232
0.9453087205];
[GH,Y, RHn] = GaussH1(@ fun,X,A)
syms x
fi1 = int(exp( -x^2).* sin(3 * x)./(sqrt(2 * pi)),x,0, + inf);
fi2 = int(exp( -x^2).* sin(3 * x)./(sqrt(2 * pi)),x,  inf,0);
fi = fi1 + fi2; Fs = double (fi), wGL = double (abs(fi - GH))
```

运行后屏幕显示取 $n=5$,用高斯－埃尔米特无穷积分公式数值计算 I 的结果 GH、截断误差公式 RHn 和精确值 Fs 及其两者的绝对误差 wGL 依次如下

```
GH =
     0
Y =
  Columns 1 through 4
   -0.08808725583239   0.08808725583239   0.10482362982925
-0.10482362982925
  Column 5
                    0
RHn =
2494507032021713/1361888527316837990400*M
Fs =
      0
wGL =
      0
```

即,取 $n=5$,用高斯－埃尔米特无穷积分公式数值计算 I 的结果 $GH=0$ 与精确值 $I\approx Fs=0$ 的绝对误差为 $wGL=0$. 根据截断误差公式(9.79)计算的截断误差的公式为

$$R_5[f,H_5]=\frac{5!\ \sqrt{\pi}}{2^n(10)!}f^{(10)}(\xi)$$

$$\approx 1.831\ 652\ 871\ 719\ 490\times 10^{-6}\cdot f^{(10)}(\xi)\ \ (-\infty<\xi<+\infty).$$

另外,根据高斯－埃尔米特无穷积分公式,编写的用 n 个节点的高斯－埃尔米特无穷积分公式计算 $\int_{-\infty}^{+\infty}\mathrm{e}^{-x^2}f(x)\,\mathrm{d}x$ 的数值积分及其估计绝对误差限的 MATLAB 程序如下:

高斯－埃尔米特无穷数值积分公式计算 $\int_{-\infty}^{+\infty}\mathrm{e}^{-x^2}f(x)\,\mathrm{d}x$ 及其误差估计的 MATLAB 主程序

输入量:*fun* 是被积函数中的因子 $f(x)$,*fun2n* 是 $f(x)$ 的 $2n$ 阶导数,$\boldsymbol{X}$ 是 n 个节点的横坐标 x_k 的 $1\times n$ 维向量,$\boldsymbol{A}$ 是系数 $A_{n,k}(k=1,2,\cdots,n)$ 的 $1\times n$ 维向量,其中 $\boldsymbol{X}$ 和 $\boldsymbol{A}$ 可以从表 9－8 中查取或自编 MATLAB 程序求得.

输出量:*Rn* 是 n 个节点的高斯－埃尔米特无穷积分公式的误差估计,*GH* 是计算 $\int_{-\infty}^{+\infty}\mathrm{e}^{-x^2}f(x)\,\mathrm{d}x$ 的数值积分值,*RHn* 是截断误差公式,$\boldsymbol{Y}$ 是 $f(x)$ 在 $\boldsymbol{X}$ 处的值.

```
function [GH,Y, RHn] = GaussH2(fun,X,A,fun2n)
n = length(X);n2 = 2 * n; Y = feval(fun,X);
GH = sum(A.* Y); sun = 1; su2n = 1; su2n1 = 1;
for k = 1:n
sun = sun * k;
end
for k = 1:n2
su2n = su2n * k;
end
mfun2n  = max(fun2n);
RHn  = ( sun * sqrt(pi)) * mfun2n /( 2^n * su2n);
```

例 9.7.10 用高斯-埃尔米特无穷积分公式计算 $\int_{-\infty}^{+\infty}\frac{1}{e^{x^2}(5+x^2)}dx$，取 $n=5$，再根据截断误差公式写出误差公式，并将计算结果与精确值进行比较.

解 (1) 建立并保存 fun.m 文件命名的 M 文件函数

```
function y  = fun(x)
          y  = 1./(5 + x.^2);
```

(2) 建立并保存 GaussH2.m 文件命名的 M 文件函数.

(3) 输入程序

```
>> syms x
fun2n = diff(exp( - x.^2)./(5 + x.^2),x,10);
fun2ns = simple(fun2n)
```

(4) 运行后再输入程序

```
>> X = [2.0201828705  -2.0201828705 0.9585724646  -0.9585724646 0];
 A  = [ 0.0199532421  0.0199532421  0.3936193232  0.3936193232
0.9453087205];
x =  - 1000:0.001:1000;
fun2ns = 32 * exp( - x.^2).* (425471484375 * x.^2 + 26348013150 *
x.^10 - 35644522500 * x.^8 - 290495953125 * x.^4 - 262342473750 * x.^6 +
13793263150 * x.^12 + 2639594700 * x.^14 + 34752150 * x.^16 - 88200525 * x.^
18 - 18517905 * x.^20 - 1466910 * x.^22 + 32520 * x.^24 + 15840 * x.^26 + 1200 *
x.^28 + 32 * x.^30 - 32395781250)./(5 + x.^2).^11;
[GH,Y, RHn] = GaussH2(@ fun,X,A, fun2ns)
syms t
fi1 = int(exp( - t.^2)./(5 + t.^2),t,0, + inf);
fi2 = int(exp( - t.^2)./(5 + t.^2),t,  - inf,0);
fi = fi1 + fi2; Fs =  double (fi), wGL  =  double (abs(fi - GH))
```

运行后屏幕显示分取 $n=5$，用高斯-埃尔米特无穷积分公式计算 I 的结果 GH

和精确值 Fs 及其两者的绝对误差 wGL 等依次如下

```
GH =
   0.32646126745168
Y =
  Columns 1 through 4
 0.11011834734512  0.11011834734512  0.16895142009535
0.16895142009535
  Column 5
  0.20000000000000
RHn =
   0.02897468774679
Fs =
   0.32640983502878
wGL =
    5.143242289669318e-005
```

取 $n=5$，用高斯-埃尔米特无穷积分公式数值计算 I 的结果 $GH=0.326\,461\,267\,451\,68$ 与精确值 $Fs=0.326\,409\,835\,028\,78$ 的绝对误差为 $wGL=5.143\,242\,289\,669\,318\text{e}-005$. 根据截断误差公式估计的绝对误差限为 $R_n=0.028\,974\,687\,746\,79$.

9.7.3 无界函数反常积分的符号计算及其 MATLAB 程序

无界函数 $f(x)$ 的反常积分 $I=\int_a^b f(x)\,\mathrm{d}x$ 有三种基本类型如下：

(1) $f(x)$ 在 $(a,b]$ 上连续，且 $\lim\limits_{x\to a^+}f(x)=\infty$ 时，若 $\lim\limits_{\varepsilon\to 0^+}\int_{a+\varepsilon}^b f(x)\,\mathrm{d}x\ (\varepsilon>0)$ 存在，则称反常积分 I 存在或收敛，且 $I=\lim\limits_{\varepsilon\to 0^+}\int_{a+\varepsilon}^b f(x)\,\mathrm{d}x$，否则反常积分 I 不存在或发散.

(2) $f(x)$ 在 $[a,b)$ 上连续，且 $\lim\limits_{x\to b^-}f(x)=\infty$ 时，若 $\lim\limits_{\varepsilon\to 0^+}\int_a^{b-\varepsilon} f(x)\,\mathrm{d}x\ (\varepsilon>0)$ 存在，则称反常积分 $\int_a^b f(x)\,\mathrm{d}x$ 存在或收敛，且 $I=\lim\limits_{\varepsilon\to 0^+}\int_a^{b-\varepsilon} f(x)\,\mathrm{d}x$，否则反常积分 $\int_a^b f(x)\,\mathrm{d}x$ 不存在或发散.

(3) $f(x)$ 在 $[a,b]$ 上除 c 点外连续，且 $\lim\limits_{x\to c}f(x)=\infty$ 时，

$$\int_a^b f(x)\,\mathrm{d}x=\int_a^c f(x)\,\mathrm{d}x+\int_c^b f(x)\,\mathrm{d}x,$$

反常积分 $\int_a^b f(x)\,\mathrm{d}x$ 收敛的充要条件是反常积分 $\int_a^c f(x)\,\mathrm{d}x$ 与 $\int_c^b f(x)\,\mathrm{d}x$ 同时收敛，且

$$\int_a^b f(x)\,\mathrm{d}x = \lim_{\varepsilon_1\to 0^+}\int_a^{c-\varepsilon_1} f(x)\,\mathrm{d}x + \lim_{\varepsilon_2\to 0^+}\int_{c+\varepsilon_2}^b f(x)\,\mathrm{d}x \ (\varepsilon_1,\varepsilon_2>0)$$

用 MATLAB 软件计算无界函数 $f(x)$ 的反常积分 $\int_a^b f(x)\,\mathrm{d}x$ 的方法有两类：一类方法是用定积分的方法计算，另一类方法是根据无界函数 $f(x)$ 的反常积分的定义计算.

例 9.7.11　求 $\int_0^1 \frac{1}{\sqrt{1-x^2}}\mathrm{d}x$.

解　(1) 输入程序

```
>> syms x
F = limit(1/(sqrt(1 -x^2)),x,1,'left')
```

运行后屏幕显示结果为

```
F =
     inf
```

即当 $x\to 1^-$ 时，被积函数 $\frac{1}{\sqrt{1-x^2}}\to\infty$.

(2) 输入程序

```
>>  syms x
F1 = int(1/(sqrt(1 -x^2)),x,0,1), LimF1 = double(F1)
```

运行后屏幕显示结果为

```
F1 -              LimF1 =
     1/2 * pi           1.5708
```

即

$$\int_0^1 \frac{1}{\sqrt{1-x^2}}\mathrm{d}x = \frac{\pi}{2}\approx 1.570\ 8.$$

例 9.7.12　求 $\int_{-1}^1 \frac{3x^5-8}{x^2}\mathrm{d}x$.

解　(1) 因为被积函数 $\frac{1}{x^2}$ 在 $[-1,1]$ 上除 $x=0$ 外连续，输入程序

```
>> syms x
F = limit((3 * x^5 -8)/(x^2),x,0)
```

运行后屏幕显示结果为

```
F =
     -inf
```

即当 $x\to 0$ 时，被积函数 $\frac{3x^5-8}{x^2}\to-\infty$.

（2）输入程序

```
>> syms x
F1 = int((3 * x^5 - 8)/(x^2),x, -1,0),
F2 = int((3 * x^5 - 8)/(x^2),x,0,1), F = F1 + F2
```

运行后屏幕显示结果为

```
F1 =            F2 =          F =
     -inf           35/4          -inf
```

即$\int_0^1 \frac{3x^5-8}{x^2}dx=\frac{35}{4}$收敛，但是$\int_{-1}^0 \frac{3x^5-8}{x^2}dx=-\infty$发散，所以按反常积分定义，有$\int_{-1}^1 \frac{3x^5-8}{x^2}dx$发散.

例 9.7.13 讨论反常积分$\int_0^1 \frac{1}{x^q}dx$ $(q=0.5,1,2)$的敛散性.

解 （1）因为被积函数$\frac{1}{x^q}$在$(0,1]$上连续，在$x=0$无定义，输入程序

```
>> syms x
LF05 = limit(1/(x^0.5),x,0,'right'),
LF1 = limit(1/x,x,0,'right')
LF2 = limit(1/(x^2),x,0,'right')
```

运行后屏幕显示结果为

```
LF05 =           LF1 =           LF2 =
      Inf             inf             inf
```

即当$q=0.5,1,2$，且$x\to 0^+$时，被积函数$\frac{1}{x^q}\to+\infty$.

（2）输入程序

```
>> syms x
F05 = int(1/(x^0.5),x,0,1), F1 = int(1/x,x,0,1),
F2 = int(1/(x^2),x,0,1)
```

运行后屏幕显示结果为

```
F05 =          F1 =          F2 =
     2              inf           inf
```

即，当$q=1$时，$\int_0^1 \frac{1}{x^q}dx=+\infty$，反常积分发散；当$q=2$时，$\int_0^1 \frac{1}{x^q}dx=+\infty$，反常积分发散；当$q=0.5$时，$\int_0^1 \frac{1}{x^q}dx=2$，反常积分收敛.

9.7.4 无界函数反常积分的近似计算及其 MATLAB 程序

我们除了可以用 MATLAB 软件中的函数 int 对无界函数反常积分进行符号计算以外，在其收敛的前提下还可以讨论其近似计算方法．计算反常积分的近似值的方法很多，例如，作变量置换法，“挖去”法，累积求和积分法等将无界函数反常积分化为有限区间上的有界函数的定积分．另外，还可以将本章前面介绍的数值积分的插值型公式、高斯型公式等推广到反常积分的近似计算．下面通过实例加以介绍．

（一）无界函数的反常积分的“挖去”法及其 MATLAB 程序

对于无界函数的反常积分 $\int_a^b f(x)\,\mathrm{d}x$（$f$ 在某点 $c\in[a,b]$ 无界）形式的积分，要“挖去” c 点附近一邻域内的积分，其关键也是要估计出这一部分的积分值．

例 9.7.14 近似计算 $I=\int_0^1 \frac{2x|\ln x|}{1+x^2}\mathrm{d}x$，误差在 10^{-6} 左右．

解 被积函数在 $x=0$ 处无界，对在小区间 $[0,r]\subset[0,1]$ 内的积分作如下估计

$$\int_0^r \frac{2x}{1+x^2}\cdot|\ln x|\mathrm{d}x\leqslant\int_0^r|\ln x|\mathrm{d}x<r(1-\ln r)=10^{-8}<10^{-6}.$$

说明：要使 I 的误差在 10^{-6} 左右，则需要 $\int_0^r \frac{2x}{1+x^2}\cdot|\ln x|\mathrm{d}x\leqslant 10^{-8}$ 或 10^{-7}．

输入程序

```
>> syms r
r = solve('r * (1 - (log(r))) = 10^( -8)',r), R = double(r)
```

运行后屏幕显示 r 的值为

```
r =
[    exp(lambertw( -1/100000000 * exp( -1)) +1)]
[ exp(lambertw(  1, -1/100000000 * exp(  1)) +1)]
R =
   2.71828181845905
   0.00000000044374
```

输入程序

```
>> r = [2.71828181845905, 0.00000000044374];
y = r.* (1 - (log(r)))
```

运行后屏幕显示 r 处的函数值为

```
y =
   1.0e - 007 *
   0.09999995049004   0.10000028042507
```

取 $r=0.000\,000\,000\,443\,74$ 时,使 $\int_0^r \frac{2x}{1+x^2}\cdot|\ln x|\mathrm{d}x<10^{-6}$,所以只需用数值方法计算 $\int_r^1 \frac{2x|\ln x|}{1+x^2}\mathrm{d}x$. 用梯形公式计算 I 的近似值 Q,输入程序

```
>> x =0.00000000044374:0.0001:1;
 y =(2 * x).* abs(log(x))./(1 +x.^2);z =trapz(x,y)
```

运行后屏幕显示

```
z =
   0.41123349222004
```

用洛巴托求积公式计算,在 MATLAB 工作区输入程序

```
>> syms x
Y =inline('(2 * x).* abs(log(x))./(1 +x.^2)');
Y1 =(2 * x).* abs(log(x))./(1 +x.^2);
[Qql,FCNTl] =quadl(Y, 0.00000000044374,1),
Q1 =int(Y1, 0.00000000044374,1), Q1d =double(Q1),
juewuQql = Q1d - Qql,
```

运行后屏幕显示

```
Qql =
   0.41123345753925
FCNTl =
   78
Q1d =
  0.41123351671206
juewuQql =
     5.917280476719355e-008
```

(二) 高斯-切比雪夫(Gauss-Chebyshev)求积公式及其 MATLAB 程序

如果在区间[-1,1]上,取 n 次切比雪夫多项式

$$T_n(x)=\cos(n\cdot\arccos x)$$

的 n 个零点 $x_k=-\cos\frac{2k-1}{2n}\pi\ (k=1,2,\cdots,n)$ 为节点,则计算在区间[-1,1]上的无界函数 $\frac{f(x)}{\sqrt{1-x^2}}$ 的反常积分的 n 个节点的高斯-切比雪夫求积公式为

$$C_n(f)=A_{n,1}f(x_1)+A_{n,2}f(x_2)+\cdots+A_{n,n}f(x_n),\tag{9.80}$$

其中

$$A_{n,k}=\int_{-1}^1\frac{T_n(x)}{(x-x_k)T'_n(x_k)\sqrt{1-x^2}}\mathrm{d}x=\frac{\pi}{n}\ (k=1,2,\cdots,n),$$

使得

$$\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}\mathrm{d}x = C_n(f) + R_n[f,C_n]. \tag{9.81}$$

因为 n 次切比雪夫多项式 $T_n(x)$ 的最高次数项的系数 $a_n = 2^{n-1}$，所以高斯－切比雪夫求积公式(9.80)的截断误差为

$$R_n[f,C_n] = \frac{f^{(2n)}(\xi)}{a_n^2(2n)!}\int_{-\infty}^{+\infty}\frac{T_n^2(x)}{\sqrt{1-x^2}}\mathrm{d}x\ (-1<\xi<1),$$

即

$$R_n[f,C_n] = \frac{2\pi}{2^{2n}(2n)!}f^{(2n)}(\xi)\ (-1<\xi<1). \tag{9.82}$$

例 9.7.15 求 $I=\int_{-1}^{1}\frac{1}{\sqrt{1-x^2}}\sum_{k=0}^{m} b_k x^k\mathrm{d}x$ 的数值公式和截断误差公式.

解 根据(9.81)式，得

$$I = \sum_{j=1}^{n}\frac{\pi}{n\cdot\sqrt{1-\left[\cos\frac{2j-1}{2n}\pi\right]^2}}\sum_{k=0}^{m}(-1)^k b_k\left[\cos\frac{2j-1}{2n}\pi\right]^k + R_n[f,C_n] \tag{9.83}$$

因为(9.83)式中的截断误差为

$$R_n[f,C_n] = \frac{2\pi}{2^{2n}(2n)!}\sum_{k=0}^{m} b_k(\xi^k)^{(2n)}(-1<\xi<1). \tag{9.84}$$

(1) 当 m 为奇数时，m 次多项式 $f(x)=\sum_{k=0}^{m}b_kx^k$ 的 $m+1$ 阶导数恒等于零，所以，取 $n=\frac{m+1}{2}$ 时，(9.84)式 $R_n[f,C_n]=0$，

$$I = \sum_{j=1}^{n}\frac{2\pi}{(m+1)\cdot\sqrt{1-\left[\cos\frac{2j-1}{m+1}\pi\right]^2}}\sum_{k=0}^{m}(-1)^k b_k\left[\cos\frac{2j-1}{m+1}\pi\right]^k.$$

(2) 当 m 为偶数时，因为 $n=\frac{m+1}{2}$ 不是正整数，所以取 $n=\frac{m+2}{2}$ 时，(9.84)式 $R_n[f,C_n]=0$，从而

$$I = \sum_{j=1}^{n}\frac{2\pi}{(m+2)\cdot\sqrt{1-\left[\cos\frac{2j-1}{m+2}\pi\right]^2}}\sum_{k=0}^{m}(-1)^k b_k\left[\cos\frac{2j-1}{m+2}\pi\right]^k.$$

根据函数 $f(x)$ 在 $[-1,1]$ 上的 n 个节点的高斯－切比雪夫求积公式编写的计算 $\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}\mathrm{d}x$ 的数值积分及其截断误差公式的 MATLAB 程序如下：

高斯 - 切比雪夫求积公式计算 $I=\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}dx$ 及其截断误差公式的 MATLAB 主程序

输入量:*fun* 是被积函数$\frac{f(x)}{\sqrt{1-x^2}}$的因式 $f(x)$,n 是节点的个数.

输出量:*GC* 是利用 n 个节点的高斯 - 切比雪夫求积公式计算 I 的数值积分值,向量 X 的元素是 n 个节点 $x_k=-\cos\frac{2k-1}{2n}\pi\ (k=1,2,\cdots,n)$,向量 Y 的元素是 $f(x)$ 在 n 个节点 $x_k(k=1,2,\cdots,n)$ 处的值,*RCn* 是截断误差公式,其中 *df2n* 是 $f(x)$ 的 $2n$ 阶导数.

```
function [GC,X,Y,RCn] = GaussC1(fun,n)
n2 = 2 * n; X = zeros(1,n); A = (pi/n) * ones(1,n);
for k = 1:n
      X(k) = -cos((2 * k -1) * pi/(2 * n));
end
      Y = feval(fun,X); GC = sum(A.* Y); su2n = 1;
for k = 1:n2
      su2n = su2n * k;
end
su2n; syms df2n,RCn = (2 * pi) * df2n/( 2^(2 * n) * su2n);
```

例 9.7.16 用高斯 - 切比雪夫求积公式计算 $I=\int_{-1}^{1}\frac{\sqrt{2+x}}{\sqrt{1-x^2}}dx$,取 $n=15$,再根据截断误差公式写出误差公式,并将计算结果与精确值进行比较.

解 (1) 建立并保存 fun.m 文件命名的 M 文件函数

```
function y = fun (x)
         y = sqrt(2 + x);
```

(2) 建立并保存 GaussC1.m 文件命名的 M 文件函数.

(3) 输入程序

```
>> n = 15; [GC,X,Y,RCn] = GaussC1(@ fun,n)
syms x
fi = int(sqrt(2 + x)./sqrt(1 - x.^2),x, -1,1); Fs = double (fi)
wGL = double (abs(fi - GC))
```

运行后屏幕显示取 $n=15$,用高斯 - 切比雪夫求积公式计算 I 的结果 *GC*、截断误差公式 *RCn* 和精确值 *Fs* 及其两者的绝对误差 *wGL*,节点 X 和 $\sqrt{2+x}$值如下

```
GC =
   4.36887628549240
```

```
X =
  Columns 1 through 9
-0.99452189536827   -0.95105651629515   -0.86602540378444
-0.74314482547739
-0.58778525229247   -0.40673664307580   -0.20791169081776
-0.00000000000000   0.20791169081776
  Columns 10 through 15
0.40673664307580    0.58778525229247    0.74314482547739
0.86602540378444   0.95105651629515   0.99452189536827
Y =
  Columns 1 through 9
1.00273531135177    1.02417941968429    1.06488243304863
1.12109552426303
1.18836641979969    1.26224536320170    1.33868902631726
1.41421356237309   1.48590433434248
  Columns 10 through 15
1.55136605708511    1.60865945814907    1.65624419258677
1.69293396320838   1.71786393998336   1.73046869239760
RCn =
1/142406544757979148169424935503213094764544 * pi * df2n
Fs =
   4.36887628549240
wGL =
    8.919048090161604e-016
```

即,取 $n=15$,用高斯-切比雪夫求积公式计算 $I \cong 4.368\ 876\ 285\ 492\ 40$,与精确值 4.368 876 285 492 40 的绝对误差为 8.919 048 090 161 60e-016. 根据截断误差公式(9.82)计算的截断误差的公式为

$$R_{15}[f,C_{15}]=\frac{2\pi}{2^{30}(30)!}f^{(30)}(\xi)$$
$$\approx 2.206\ 073\ 224\ 323\ 33\times 10^{-41}\cdot f^{(30)}(\xi)\quad(-\infty<\xi<+\infty).$$

另外,根据函数 $f(x)$ 在 $[-1,1]$ 上的 n 个节点的高斯-切比雪夫求积公式,还可编写计算 $\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}\mathrm{d}x$ 的数值积分及其估计绝对误差限的 MATLAB 程序如下:

n 个节点的高斯 - 切比雪夫求积公式计算 $\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}dx$ 及其误差估计的 MATLAB 主程序

输入量：fun 是被积函数 $\frac{f(x)}{\sqrt{1-x^2}}$ 的因式 $f(x)$，n 是节点的个数，$df2n$ 是 $f(x)$ 的 $2n$ 阶导数.

输出量：RCn 是截断误差估计，GC 是利用 n 个节点的高斯 - 切比雪夫求积公式计算 $\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}dx$ 的数值积分值，向量 X 的元素是 n 个节点 $x_k=-\cos\frac{2k-1}{2n}\pi\ (k=1,2,\cdots,n)$，向量 Y 的元素是 $f(x)$ 在 n 个节点 $x_k(k=1,2,\cdots,n)$ 处的值.

```
function [GC, X, Y, RCn] =GaussC2(fun,n ,df2n)
n2 =2 * n; X =zeros(1,n); A =(pi/n) * ones(1,n);
for k =1:n
  X(k) = -cos((2 * k -1) * pi/(2 * n));
end
  Y =feval(fun,X); GC =sum(A. * Y); su2n =1;
for k =1:n2
  su2n =su2n * k;
end
su2n; mfun2n =max(df2n); RCn = (2 * pi) * mfun2n /( 2^(2 * n) * su2n);
```

例 9.7.17 用高斯 - 切比雪夫求积公式计算 $I=\int_{-1}^{1}\frac{e^{-x^2}}{\sqrt{1-x^2}}dx$，取 $n=6$，估计其误差，并将计算结果与精确值进行比较.

解 (1) 建立并保存 fun.m 文件命名的 M 文件函数

```
function y = fun(x)
         y = exp( -x.^2);
```

(2) 建立并保存 GaussC2 .m 文件命名的 M 文件函数.

(3) 输入程序

```
>> syms x
fun2n =diff(exp( -x.^2),x,12); df2n =simple(fun2n)
```

(4) 运行后再输入程序

```
>> x = -1:0.0001:1;
df2n =64 * exp( -x.^2). * (10395 -124740 * x.^2 +207900 * x.^4 -
```

```
110880*x.^6+23760*x.^8-2112*x.^10+64*x.^12);
        n=6;[GC, X, Y, RCn]=GaussC2 (@ fun,n ,df2n)
        syms t a b
        fi1=int(exp(-t.^2)./sqrt(1-t.^2),t,0,a); fia=limit(fi1,a,1)
        fi2=int(exp(-t.^2)./sqrt(1-t.^2),t, -b,0); fib=limit(fi2,
b,1), fi=fia+fib;
        Fs = double (fi), wGL = double (abs(fi - GC))
```

运行后屏幕显示结果依次如下

```
        GC =
          2.02643676313532
        X =
          Columns 1 through 4
         -0.96592582628907   -0.70710678118655   -0.25881904510252
0.25881904510252
          Columns 5 through 6
         0.70710678118655   0.96592582628907
        Y =
          Columns 1 through 4
          0.39336682642326    0.60653065971263    0.93520708016087
0.93520708016087
          Columns 5 through 6
           0.60653065971263   0.39336682642326
        RCn =
            2.130528872063391e-006
        Warning: Explicit integral could not be found.
        fia =
            1/2*pi*hypergeom([1/2],[1],-1)
        Warning: Explicit integral could not be found.
        fib =
            int(exp(-t^2)/(1-t^2)^(1/2),t = -1..0)
        Fs =                            wGL =
           2.02643806694936                1.303814033355041e-006
```

即,取 $n=6$,用高斯-切比雪夫求积公式计算 $I\approx 2.026\,436\,763\,135\,32$ 与精确值 $Fs=2.026\,438\,066\,949\,36$ 的绝对误差为 $wGL=1.303\,814\,033\,355\,04\text{e}-006$. 根据截断误差公式估计的误差为 $RCn=2.130\,528\,872\,063\,39\text{e}-006$.



习 题 9.7

1. 近似计算 $\int_0^1 \frac{|\ln x|}{1+x^3}\mathrm{d}x$,误差在 10^{-6}左右.

2. 分别用三点高斯-勒让德积分公式、步长为3的梯形公式和步长为2的辛普森求积公式计算 $\int_{-5}^0 \frac{1}{5+x}\mathrm{d}x$,并将计算结果与精确值进行比较.

3. 近似计算 $I=\int_0^{+\infty} \mathrm{e}^{-x^2}\mathrm{d}x$,使精确度为 10^{-12}.

4. 近似计算 $I=\int_0^{+\infty} \mathrm{e}^{x^3}\cos 2x\mathrm{d}x$,取精度分别为 10^{-14}和 10^{-4},并与精确值比较.

用 MATLAB 软件完成下列问题,并用一个文件名存盘.

5. 下列反常积分中收敛的是(　　).

A. $\int_1^{+\infty} \ln x\mathrm{d}x$　　B. $\int_1^{+\infty} \frac{1}{x}\mathrm{d}x$　　C. $\int_1^{+\infty} \frac{1}{x^2}\mathrm{d}x$　　D. $\int_1^{+\infty} \mathrm{e}^x\mathrm{d}x$

6. 下述结论错误的是(　　).

A. $\int_0^{+\infty} \frac{x}{1+x^2}\mathrm{d}x$ 发散　　B. $\int_0^{+\infty} \frac{1}{1+x^2}\mathrm{d}x$ 收敛

C. $\int_{-\infty}^{+\infty} \frac{x}{1+x^2}\mathrm{d}x=0$　　D. $\int_{-\infty}^{+\infty} \frac{x}{1+x^2}\mathrm{d}x$ 发散

7. 讨论下列无穷区间上的反常积分的敛散性.

(1) $\int_2^{+\infty} \frac{7x^p}{x^4+1}\mathrm{d}x$ $(p=1,2,3,8)$;　　(2) $\int_{-\infty}^{-1} \frac{1}{x^{4p}}\mathrm{d}x$ $(p=-2,0.25,0.5,1,4)$;

(3) $\int_{-\infty}^{+\infty} \frac{1}{4+9x^2}\mathrm{d}x$;　　(4) $\int_{-\infty}^{+\infty} \frac{1}{x^2+2x+2}\mathrm{d}x$.

8. 讨论下列反常积分的敛散性.

(1) $\int_{-1}^0 \frac{1}{\sqrt{1-x^2}}\mathrm{d}x$;　　(2) $\int_1^{\mathrm{e}} \frac{1}{x\sqrt{1-(\ln x)^2}}\mathrm{d}x$;

(3) $\int_1^{+\infty} \frac{1}{x\ln^2 x}\mathrm{d}x$;　　(4) $\int_0^2 \frac{1}{(1-x)^2}\mathrm{d}x$.

9. 讨论下列反常积分的敛散性.

(1) $\int_0^{+\infty} \frac{x}{1-x^2}\mathrm{d}x$;　　(2) $\int_0^{+\infty} \frac{1}{\sqrt{x}}\mathrm{d}x$;

(3) $\int_a^{+\infty} \frac{1}{(x-a)^q}\mathrm{d}x$ $(q=0.3,1,3)$;　　(4) $\int_{-\infty}^0 \frac{1}{x^2+2x+1}\mathrm{d}x$.

10. 一椭球的三个半轴的长度分别为4,3,2,求它的表面积.

11. 近似计算 $I=\int_0^{+\infty} \mathrm{e}^{-5x^2}\mathrm{d}x$,使精确度为 10^{-4}.

12. 近似计算 $\int_0^1 \frac{|\ln x|}{1+x}dx$,使精确度为 10^{-6}. （提示：用梯形公式可得结果为 0.822 343,而精确值是 $\int_0^1 \frac{|\ln x|}{1+x}dx = \frac{\pi^2}{12} \approx 0.822\ 467$).

13. 用数值方法计算 $I = \int_0^{+\infty} \frac{e^{-x}}{1+x^4}dx$,使精确度为 10^{-6},并与精确值比较.

14. 如果取 $r_k = 2k\pi, k = 0,1,2,\cdots$,根据(9.72)式,当 k 取何值时,$I = \int_0^{+\infty} \frac{\cos x}{1+x^2}dx$ 的近似值的截断误差为 10^{-6},计算其近似值,并与精确值比较.

15. 用高斯－拉盖尔无穷积分公式计算 $I = \frac{1}{\sqrt{2\pi}}\int_0^{+\infty} e^{-x}\cos 5x dx$,取 $n=5$,再根据截断误差公式写出误差公式,并将计算结果与精确值进行比较.

16. 用高斯－拉盖尔无穷积分公式计算 $\int_0^{+\infty} \frac{e^{-x}}{7+x^2}dx$,取 $n=6$,再根据截断误差公式写出误差公式,并将计算结果与精确值进行比较.

17. 用高斯－埃尔米特无穷积分公式计算 $I = \int_{-\infty}^{+\infty} e^{-x^2}\cos 7x dx$,取 $n=4$,再根据截断误差公式写出误差公式,并将计算结果与精确值进行比较.

18. 用高斯－埃尔米特无穷积分公式计算 $\int_{-\infty}^{+\infty} \frac{e^{-x^2}}{9+3x^2}dx$,取 $n=6$,再根据截断误差公式写出误差公式,并将计算结果与精确值进行比较.

19. 用高斯－切比雪夫求积公式计算 $I = \int_{-1}^{1} \frac{\sqrt{7x+9}}{\sqrt{1-x^2}}dx$,取 $n=14$,再根据截断误差公式写出误差公式,并将计算结果与精确值进行比较.

20. 用高斯－切比雪夫求积公式计算 $\int_{-1}^{1} \frac{e^{\sin x^2}}{\sqrt{1-x^2}}dx$,取 $n=9$,估计其误差,并将计算结果与精确值进行比较.

9.8 多重积分的计算及其 MATLAB 程序

将本章前面各节讨论的数值积分和符号积分的方法稍加修改,便可用来计算重积分.本节主要介绍二重积分和三重积分的符号计算和数值计算及其 MATLAB 程序.

9.8.1 二重积分的符号计算及其 MATLAB 程序

为了将多重积分的数值计算与对应的精确解比较,我们首先介绍用软件 MATLAB 做二重积分的符号计算的方法.

因为二重积分可以转化为二次积分运算,即

$$\iint\limits_{D_{xy}} f(x,y)\,\mathrm{d}\sigma = \int_a^b \mathrm{d}x \int_{y_1(x)}^{y_2(x)} f(x,y)\,\mathrm{d}y,$$

或

$$\iint\limits_{D_{xy}} f(x,y)\,\mathrm{d}\sigma = \int_c^d \mathrm{d}y \int_{x_1(y)}^{x_2(y)} f(x,y)\,\mathrm{d}x.$$

所以,我们可以用 MATLAB 函数 int 计算两个定积分的方法计算二次积分.具体步骤参考下面的实例.

例 9.8.1 计算 $\iint\limits_{D_{xy}} \cos(x+y)\,\mathrm{d}\sigma$,其中 D_{xy}是由曲线 $2xy=1$,$y=\sqrt{2x}$,$x=2.5$ 所围成的平面区域.

解 (1) 画出积分区域的草图.输入程序

```
>> x = 0.001:0.001:3; y1 = 1./(2 * x);y2 = sqrt(2 * x);
plot(x,y1,'b-',x,y2,'M',2.5,y,'r-'), axis([-0.5 3 -0.5 3])
title('由 y1 = 1/(2x),y2 = sqrt(2x)和 x = 2.5 所围成的积分区域 Dxy')
```

运行后屏幕显示图 9-6.

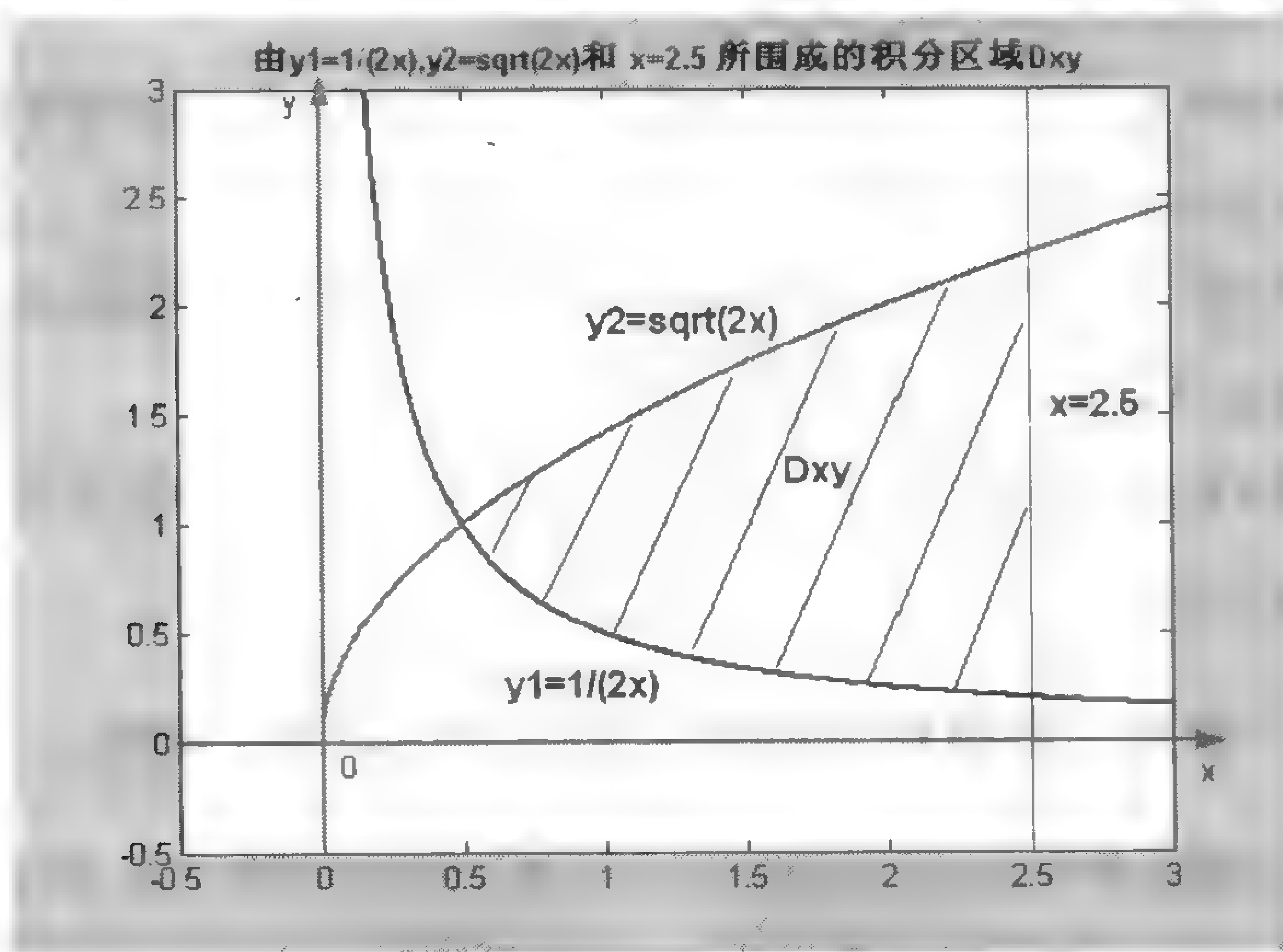


图 9-6 由 $2xy=1$,$y=\sqrt{2x}$,$x=2.5$ 所围成的积分区域 D_{xy}

(2) 确定积分限.输入程序

```
>> syms x y
y1 = ('2 * x * y = 1'); y2 = ('y - sqrt(2 * x) = 0');
[x,y] = solve(y1,y2,x,y)
```

运行后屏幕显示两条曲线 $2xy=1$,$y=\sqrt{2x}$的交点如下

```
x =1/2,y =1
```

(3) 输入计算程序

```
>> syms x y
f =cos(x +y);y1 =1/(2 * x);y2 =sqrt(2 * x); jfy =int(f,y,y1,y2);
jfx =int(jfy,x,0.5,2.5); jf2 =double(jfx)
```

运行后屏幕显示如下

```
Warning: Explicit integral could not be found.
jf2 =
  -1.83209375329577
```

因此,所求的 $\iint\limits_{D_{xy}} \cos(x+y)\,\mathrm{d}\sigma$ 的近似值为 $-1.832\,093\,753\,295\,77$.

例 9.8.2 计算 $\iint\limits_{D_{xy}} \frac{\sin(x+y)}{x+y}\mathrm{d}\sigma$,其中 D_{xy} 是由曲线 $x=y^2, y=x-2$ 所围成的平面区域.

解 (1) 画出积分区域的草图.输入程序

```
>> syms x y
f1 =x -y^2;f2 =x -y -2; ezplot(f1),hold on
ezplot(f2),hold off,axis([ -0.5 5 -1.5 3])
title('由 x =y^2 和 y =x -2 所围成的积分区域 Dxy ')
```

运行后屏幕显示图 9-7.

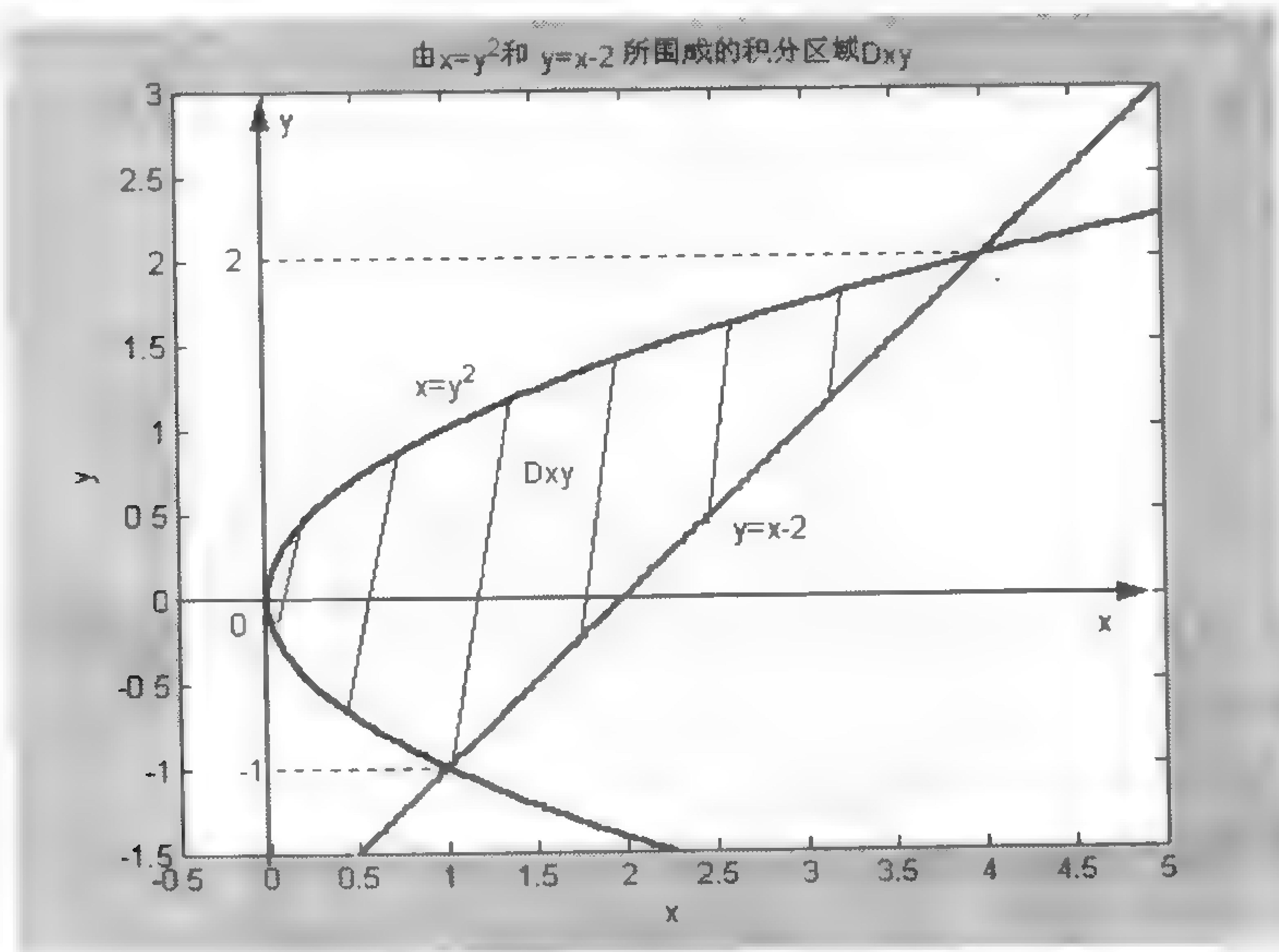


图 9-7 由两条曲线 $x=y^2, y=x-2$ 所围成的积分区域 D_{xy}

(2) 确定积分限. 输入程序

```
>> syms x y
y1 = ('x - y^2 = 0'); y2 = ('x - y - 2 = 0'); [x,y] = solve(y1,y2,x,y)
```

运行后屏幕显示两条曲线 $x = y^2, y = x - 2$ 的交点如下

```
x =                    y =
      [1]                   [-1]
      [4]                   [ 2]
```

(3) 输入计算程序

```
>> syms x y
f = sin(x + y)/(x + y);x1 = y^2;x2 = y + 2; jfx = int(f,x,x1,x2);
jfy = int(jfx,y, -1,2); jf2 = double(jfy)
```

运行后屏幕显示如下

```
Warning: Explicit integral could not be found.
jf2 =
    1.97124962844910
```

因此,所求的 $\iint\limits_{D_{xy}} \frac{\sin(x+y)}{x+y} d\sigma \approx 1.971\ 249\ 628\ 449\ 10.$

9.8.2 二重积分的梯形公式及其 MATLAB 程序

数值计算二重积分的方法很多,这里主要介绍将二重积分转化为二次积分运算的数值方法. 如果二重积分 $\iint\limits_{D_{xy}} f(x,y)d\sigma$ 的积分区域 D_{xy}是矩形区域 $a \leq x \leq b, c \leq y \leq d$,则可以将二重积分转化为二次积分运算,即

$$I(f) = \iint\limits_{D_{xy}} f(x,y)d\sigma = \int_a^b dx \int_c^d f(x,y)dy, \tag{9.85}$$

或

$$\iint\limits_{D_{xy}} f(x,y)d\sigma = \int_c^d dy \int_a^b f(x,y)dx. \tag{9.86}$$

这样,我们可以用本章介绍的数值积分的各种方法计算两个定积分即可. 下面介绍如何将梯形公式推广到数值计算二次积分及其 MATLAB 程序.

(一) 二重积分的基本梯形公式及其 MATLAB 程序

利用基本梯形公式,将(9.86)式中的第一层定积分化为

$$\int_a^b f(x,y)dx \approx \frac{(b-a)\left[f(a,y)+f(b,y)\right]}{2}. \tag{9.87}$$

将(9.87) 式代入(9.86) 式,得

$$\iint_{D_{xy}} f(x,y)\,\mathrm{d}\sigma = \int_c^d \mathrm{d}y\int_a^b f(x,y)\,\mathrm{d}x \approx \frac{b-a}{2}\left[\int_c^d f(a,y)\,\mathrm{d}y + \int_c^d f(b,y)\,\mathrm{d}y\right].$$

再利用基本梯形公式(9.28),得

$$\iint_{D_{xy}} f(x,y)\,\mathrm{d}\sigma \approx \frac{(b-a)(d-c)}{4}[f(a,c)+f(a,d)+f(b,c)+f(b,d)], \tag{9.88}$$

称(9.88)式为数值计算二重积分 $I(f)$ 的**基本梯形公式**.

例 9.8.3 根据基本梯形公式编写 MATLAB 程序,计算 $I=\iint_{D_{xy}} \frac{\sin(x+y)}{x+y}\mathrm{d}\sigma$ 的近似值,并将计算结果与精确值比较. 其中 D_{xy} 是矩形区域 $0\leqslant x\leqslant 4$, $-1\leqslant y\leqslant 2$.

解 根据(9.88)式编写并输入下列 MATLAB 程序

```
>> a=0;b=4;c=-1;d=2; fac=sin(a+c)/(a+c);fad=sin(a+
d)/(a+d);
fbc=sin(b+c)/(b+c);fbd=sin(b+d)/(b+d); h=(b-a)*(d-c);
F=fac+fad+fbc+fbd; T2=(h*F)/4
syms x y
bjh=sin(x+y)/(x+y); jfx=int(bjh,x,a,b);
jfy=int(jfx,y,c,d); I2=double(jfy), juewu=abs(I2-T2)
```

运行后屏幕显示二重积分 I 的精确值 I_2,根据(9.88)式计算 I 的近似值 T_2,近似值 T_2 与精确值 I_2 的绝对误差 *juewu* 如下

```
T2 =                  I2 =                  juewu =
  3.88977135362262      3.64403676611252      0.24573458751009
```

(二) 二重积分的复合梯形公式及其 MATLAB 程序

为了提高精度,用两组等距平行直线族

$$x_k = a + kh \ \left(k=0,1,2,\cdots,n, h_x=\frac{b-a}{n}\right)$$

$$y_j = c + jh \ \left(j=0,1,2,\cdots,m, h_y=\frac{d-c}{m}\right), \tag{9.89}$$

将积分区域

$$D_{xy}=\{(x,y)\mid a\leqslant x\leqslant b, c\leqslant y\leqslant d\} \tag{9.90}$$

分割成 $n\times m$ 个子矩形域

$$D_{kj}=\{(x,y)\mid x_k\leqslant x\leqslant x_{k+1}, y_j\leqslant y\leqslant y_{j+1}\}$$
$$(k=0,1,2,\cdots,n-1, j=0,1,2,\cdots,m-1), \tag{9.91}$$

则称两组等距直线族的交点 (x_k,y_j) $(k=0,1,2,\cdots,n-1,j=0,1,2,\cdots,m-1)$ 为**求积节点**. 如果在每个子矩形域 D_{kj} 上应用数值计算二重积分 $I(f)$ 的基本梯形公式(9.87),便得到如下定理.

定理 9.8(二重积分复合梯形公式) 如果函数$f(x,y)$在积分区域(9.90)上的$n\times m$个子矩形域(9.91)的交点(x_k,y_j) $(k=0,1,2,\cdots,n-1,j=0,1,2,\cdots,m-1)$处有定义,且其函数值为$f(x_k,y_j)$,其中$(x_k,y_j)$满足(9.89),则在积分区域$D_{xy}$上有

$$I(f)=\iint\limits_{D_{xy}}f(x,y)\,\mathrm{d}\sigma=\sum_{k=0}^{n-1}\sum_{j=0}^{m-1}\iint\limits_{D_{kj}}f(x,y)\,\mathrm{d}x\mathrm{d}y$$

$$\approx\frac{h_xh_y}{4}\sum_{k=0}^{n-1}\sum_{j=0}^{m-1}\left[f(x_k,y_j)+f(x_k,y_{j+1})+f(x_{k+1},y_j)+f(x_{k+1},y_{j+1})\right] \tag{9.92}$$

$$=\frac{h_xh_y}{4}\sum_{k=0}^{n-1}\sum_{j=0}^{m-1}p_{kj}f(x_k,y_j), \tag{9.93}$$

其中

$$\begin{cases}p_{00}=p_{0m}=p_{n0}=p_{nm}=1,\\ p_{k0}=p_{km}=2, \quad k=1,2,\cdots,n-1,\\ p_{0j}=p_{nj}=2, \quad j=1,2,\cdots,m-1,\\ p_{kj}=4, \qquad k=1,2,\cdots,n-1,j=1,2,\cdots,m-1,\end{cases}$$

称(9.93)式为**数值计算二重积分$I(f)$的复合梯形公式**.

例 9.8.4 根据(9.93)式编写 MATLAB 程序计算$I=\iint\limits_{D_{xy}}\frac{\mathrm{e}^{-(x^2+y^2)}}{x+y}\mathrm{d}\sigma$近似值,分别取$(n,m)=(2,2),(12,12),(112,112),(1112,1112)$,并将计算结果与精确值比较. 观察节点的个数与误差的关系. 问取$(n,m)=\{(111112,111112)\}$时,计算机能计算吗,为什么? 其中$D_{xy}$是矩形区域$0\leqslant x\leqslant 4,1\leqslant y\leqslant 2$.

解 (1) 根据(9.93)式编写下列 MATLAB 主程序并保存名为r2trapf2.m 的 M 文件

```
function [T2,I2,juewu] = r2trapf2(a,b,n,c,d,m)
x = zeros(1,n);y = zeros(1,m); P = zeros(n,m); fyx = zeros(n,m);
hx = (b - a) / (n - 1);hy = (d - c) / (m - 1); x(1) = a;x(n) = b;y(1) = c;
y(m) = d;
for k = 1:n - 1
x(k + 1) = a + k * hx;
    for j = 1:m - 1
    y(j + 1) = c + j * hy;
    fyx(k + 1,j + 1) = exp( - (x(k + 1)^2 + y(j)^2)) / (x(k + 1) + y(j));
    fyx(1,j + 1) = exp( - (x(1)^2 + y(j + 1)^2)) / (x(1) + y(j + 1));
```

```
        fyx(k+1,1)=exp(-(x(k+1)^2+y(1)^2))/(x(k+1)+y(1));
        fyx(1,1)=exp(-(x(1)^2+y(1)^2))/(x(1)+y(1));
         fyx(1,m)=exp(-(x(1)^2+y(m)^2))/(x(1)+y(m));
        fyx(n,1)=exp(-(x(n)^2+y(1)^2))/(x(n)+y(1));
        fyx(n,m)=exp(-(x(n)^2+y(m)^2))/(x(n)+y(m));
        P(k+1,j+1)=4;P(1,j+1)=2;P(n,j+1)=2;
        P(k+1,1)=2; P(k+1,m)=2;
        P(1,1)=1; P(1,m)=1; P(n,1)=1; P(n,m)=1;
        end
    end
    P;fyx; F=P.*fyx; F1=sum(F); F2=sum(F1); T2=hx*hy*F2/4;
    syms t w
    bjh=exp(-(w^2+t^2))/(w+t); jfx=int(bjh,w,a,b);
    jfy=int(jfx,t,c,d); I2=double(jfy); juewu=abs(I2-T2);
```

(2) 输入下列 MATLAB 程序

```
>> a=0;b=4;n2=2;c=1;d=2;m2=2;[T2,I2,juewu2]=r2trapf2
(a,b,n2,c,d,m2)
n12=12;;m12=12;[T12,I12,juewu12]=r2trapf2(a,b,n2,c,d,m12)
n112=112;;m112=112;[T112,I2,juewu112]=r2trapf2(a,b,n112,c,
d,m112)
n1112=1112;
m1112=1112;[T1112,I2,juewu1112]-r2trapf2(a,b,n1112,c,d,
m1112)
```

运行后屏幕显示分别取$(n,m)=(2,2),(12,12),(112,112),(1112,1112)$时，用(9.93)式编写 MATLAB 程序计算二重积分的值 $T_2,T_{12},T_{112},T_{1112}$，精确值的近似值 I_2，及其误差 $juewu_2,juewu_{12},juewu_{112},juewu_{1112}$结果如下

```
Warning: Explicit integral could not be found.
T2 =                    I2 =                   juewu2 =
   0.37703726923921       0.06889208904324        0.30814518019597
T12 =                  juewu12 =               T112 =
  0.21706671440746       0.14817462536422         0.07079360039872
juewu112 =             T1112 =                juewu1112 =
  0.00190151135548     0.06908616401326     1.940749700208577e-004
```

由此可见，节点的个数越多，近似值与精确值的绝对误差越小. 但是，取$(n,m)=\{(111\ 112,\ 111\ 112)\}$时，计算机就不能计算了，因为计算机内存不够. 见下例：

```
>> a=0;b=4;n2=111112;c=1;d=2;m2=111112;
[T2,I2,juewu2]=r2trapf2(a,b,n2,c,d,m2)
```

```
??? Error using = = > zeros
Product of dimensions is greater than maximum integer.

Error in ==> D:\MATLAB6p5\work\r2trapf2.m
On line 2  ==> x=zeros(1,n);y=zeros(1,m); P=zeros(n,m); fyx
=zeros(n,m);
```

根据复合梯形公式(9.93)式编写了如下数值计算 $\iint\limits_{D_{xy}} f(x,y)\,\mathrm{d}\sigma$ 的近似值 MATLAB 主程序. 数值计算调用前,首先要建立被积函数 $f(x,y)$ 的 M 函数文件或指定被积函数 $f(x,y)$ 为嵌入对象 F = inline('fun'),然后调用.

用复合梯形公式(9.93)数值计算二重积分 $I_2=\iint\limits_{D_{xy}} f(x,y)\,\mathrm{d}\sigma$ 的MATLAB主程序

输入量:*fun* 是被积函数 $f(x,y)$,a,b,c,d 分别是积分变量 x,y 的积分下、上限,n 和 m 表示将积分区域 $D_{xy}=\{(x,y)\mid a\leqslant x\leqslant b,c\leqslant y\leqslant d\}$ 分割成 $n\times m$ 个子矩形域.

输出量:T_2 是利用复合梯形公式(9.93)数值计算 I_2 的近似值,I_2 是二重积分 I_2 的精确值,*juewu* 是精确值 I_2 与近似值 T_2 的绝对误差.

```
function [T2,I2,juewu] = r2trapf2(fun,a,b,n,c,d,m)
x = zeros(1,n);y = zeros(1,m); P = zeros(n,m); fyx = zeros(n,m);
hx = (b - a)/(n - 1);hy = (d - c)/(m - 1); x(1) = a;x(n) = b;y(1) = c;
y(m) = d;
for k = 1:n - 1
x(k + 1) = a + k * hx;
    for j = 1:m
    y(j + 1) = c + j * hy;
    fyx(k + 1,j + 1) = feval(fun,x(k + 1),y(j));
    fyx(1,j + 1) = feval(fun,x(1),y(j + 1));
    fyx(k + 1,1) = feval(fun,x(k + 1),y(1));
    fyx(1,1) = feval(fun,x(1),y(1));
    fyx(1,m) = feval(fun,x(1),y(m));
    fyx(n,1) = feval(fun,x(n),y(1));
    fyx(n,m) = feval(fun,x(n),y(m));
    P(k + 1,j + 1) = 4; P(1,j + 1) = 2;P(n,j + 1) = 2; P(k + 1,1) = 2;
    P(k + 1,m) = 2;
    P(1,1) = 1; P(1,m) = 1; P(n,1) = 1; P(n,m) = 1;
```

```
    end
end
P;fyx; F-P.*fyx; F1=sum(F); F2=sum(F1); T2=hx*hy*F2/4;
syms t w
bjh=feval(fun,w,t); jfx=int(bjh,w,a,b); jfy=int(jfx,t,c,d);
I2=double(jfy); juewu=abs(I2-T2);
```

9.8.3 矩形域上的辛普森公式及其 MATLAB 程序

我们可以利用将定积分的梯形公式推广到二重积分的方法,将定积分的辛普森公式推广到二重积分.下面我们从另一个角度,按照二次积分的计算次序,用定积分复合辛普森公式推导出二重积分的复合辛普森公式的方法.

(一) 矩形域上的复合辛普森公式及其误差分析

考虑计算(9.85)式,即

$$I(f)=\iint_{D_{xy}} f(x,y)\,\mathrm{d}\sigma=\int_a^b \mathrm{d}x\int_c^d f(x,y)\,\mathrm{d}y.$$

首先用两组等距平行直线族

$$x_k=a+kh_x\left(k=0,1,2,\cdots,2n,h_x=\frac{b-a}{2n}\right),$$

$$y_j=c+jh_y\left(j=0,1,2,\cdots,2m,h_y=\frac{d-c}{2m}\right). \tag{9.94}$$

将积分区域(9.90)分割成$(2n)\times(2m)$个子矩形域

$$D_{kj}=\{(x,y)\mid x_k\leqslant x\leqslant x_{k+1},y_j\leqslant y\leqslant y_{j+1}\}, \tag{9.95}$$

$k=0,1,2,\cdots,2n-1,j=0,1,2,\cdots,2m-1$. 先计算定积分

$$F(x)=\int_c^d f(x,y)\,\mathrm{d}y. \tag{9.96}$$

将 x 看作常数,用定积分复合辛普森公式(9.11)计算(9.96)式,得

$$F(x)=\frac{h_y}{3}\left[f(x,c)+f(x,d)+2\sum_{j=1}^{m-1}f(x,y_{2j})+4\sum_{j=0}^{m-1}f(x,y_{2j+1})\right]+R(F,S_m), \tag{9.97}$$

其中的截断误差 $E(F,S_m)$为

$$R(F,S_m)=-\frac{h_y^4(d-c)}{180}\cdot\frac{\partial^4 f(x,\xi)}{\partial y^4},\xi\in[c,d],$$

将(9.97)式代入(9.85)式,得

$$I(f)=\frac{h_y}{3}\Big[\int_a^b f(x,c)\,\mathrm{d}x+\int_a^b f(x,d)\,\mathrm{d}x+2\sum_{j=1}^{m-1}\int_a^b f(x,y_{2j})\,\mathrm{d}x+$$
$$4\sum_{j=0}^{m-1}\int_a^b f(x,y_{2j+1})\,\mathrm{d}x\Big]+\int_a^b E(F,S_m)\,\mathrm{d}x, \tag{9.98}$$

在节点 $x_k = a + kh_x\left(k=0,1,2,\cdots,2n, h_x=\dfrac{b-a}{2n}\right)$处，对每一个 $j=0,1,2,\cdots,2m$ 都有

$$
\begin{aligned}
I(f(x,y_j)) &= \int_a^b f(x,y_j)\,\mathrm{d}x \\
&= \frac{h_x}{3}\Big[f(a,y_j)+f(b,y_j)+2\sum_{k=1}^{n-1}f(x_{2k},y_j)+ \\
&\quad 4\sum_{k=0}^{n-1}f(x_{2k+1},y_j)\Big]-\frac{(b-a)h_x^4}{180}\cdot\frac{\partial^4 f(\eta_j,y_j)}{\partial x^4},\eta_j\in[a,b]
\end{aligned}
$$

作和，便得到如下定理.

定理 9.9（二重积分复合辛普森公式） 如果函数 $f(x,y)$ 在积分区域(9.90)上的 $2n\times 2m$ 个子矩形域(9.95)的交点 (x_k,y_j) $(k=0,1,2,\cdots,n-1,j=0,1,2,\cdots,m-1)$ 处有定义，且其函数值为 $f(x_k,y_j)$，其中 (x_k,y_j) 满足(9.94)，则在积分区域 D_{xy} 上存在**二重积分复合梯形公式**

$$
\begin{aligned}
S_{2n\times 2m} = \frac{h_xh_y}{9}\Big[&f(a,c)+f(b,c)+f(a;d)+f(b,d)+2\sum_{k=1}^{n-1}f(x_{2k},c)+2\sum_{j=1}^{m-1}f(b,y_{2j})+ \\
&2\sum_{j=1}^{m-1}f(a,y_{2j})+2\sum_{k=1}^{n-1}f(x_{2k},d)+4\sum_{k=0}^{n-1}f(x_{2k+1},c)+4\sum_{j=0}^{m-1}f(a,y_{2j+1})+ \\
&4\sum_{j=0}^{m-1}f(b,y_{2j+1})+4\sum_{k=0}^{n-1}f(x_{2k+1},d)+ \\
&4\sum_{j=1}^{m-1}\sum_{k=1}^{n-1}f(x_{2k},y_{2j})+8\sum_{j=1}^{m-1}\sum_{k=0}^{n-1}f(x_{2k+1},y_{2j})+ \\
&8\sum_{j=0}^{m-1}\sum_{k=1}^{n-1}f(x_{2k},y_{2j+1})+16\sum_{j=0}^{m-1}\sum_{k=0}^{n-1}f(x_{2k+1},y_{2j+1})\Big],
\end{aligned}
\tag{9.99}
$$

使得

$$
\iint_{D_{xy}} f(x,y)\,\mathrm{d}\sigma = S_{2n\times 2m}+E(f,S_{2n\times 2m}). \tag{9.100}
$$

推论 9.7（二重积分复合辛普森公式的误差分析） 如果被积函数 $f(x,y)$ 在积分区域(9.90)上具有连续的四阶偏导数，则**二重积分复合辛普森公式**(9.99)式的截断误差，即(9.100)式中的 $E(f,S_{2n\times 2m})$ 为

$$
E(f,S_{2n\times 2m}) = \frac{(a-b)(d-c)}{180}\left[h_x^4\frac{\partial^4 f(\bar{\eta},\bar{\xi})}{\partial x^4}+h_y^4\frac{\partial^4 f(\tilde{\eta},\tilde{\xi})}{\partial y^4}\right], \tag{9.101}
$$

其中 $(\bar{\eta},\bar{\xi}),(\tilde{\eta},\tilde{\xi})\in\{(x,y)\mid a\leqslant x\leqslant b,c\leqslant y\leqslant d\}$.

（二）复合辛普森公式的 MATLAB 程序

计算机软件 MATLAB 系统提供了数值计算二重积分 $\iint\limits_{D_{xy}} f(x,y)\,\mathrm{d}x\mathrm{d}y$ 的函数

dblquad,其中积分区域 $D_{xy}=\{(x,y)\mid a\leqslant x\leqslant b,c\leqslant y\leqslant d\}$ 是矩形域,它的具体调用格式如下:

调用格式一:Q2 = dblquad(FUN,a,b,c,d)

输入量:*FUN* 是被积函数 $f(x,y)$,a,b,c,d 是矩形区域 D_{xy} 中 $a\leqslant x\leqslant b$, $c\leqslant y\leqslant d$.

输出量:Q_2是用自适应辛普森公式和 MATLAB 函数 quad 计算二重积分 $\iint\limits_{D_{xy}} f(x,y)\mathrm{d}x\mathrm{d}y$ 的估计值,其误差限为 10^{-6}.

调用格式二:Q2 = dblquad(FUN,a,b,c,d,tol)

同上,但指定绝对误差限为 *tol*,默认值为 10^{-6}.

调用格式三:Q2 = dblquad(FUN,a,b,c,d,tol,@QUADL)

指定用 MATLAB 函数 quadl 代替函数 quad 计算二重积分 $\iint\limits_{D_{xy}} f(x,y)\mathrm{d}x\mathrm{d}y$ 的估计值,默认值是函数 quad.其他同上.

调用格式四:Q2 = dblquad(FUN,a,b,c,d,tol,@MYQUADF)

指定使用用户自己编写的求二重积分的 MATLAB 程序 MYQUADF.m 代替函数 quad 计算二重积分 $\iint\limits_{D_{xy}} f(x,y)\mathrm{d}x\mathrm{d}y$ 的估计值,其中程序 MYQUADF.m 的调用顺序应该与 quadl 和 quad 的相同,默认值是函数 quad.其他同上.

调用格式五:Q2 = dblquad(FUN,a,b,c,d,tol,@ QUADL,P1,P2,...)

此命令规定对函数 fun(x,y,P1,P2,...)附加的参数 $P_1,P_2,\cdots$ 直接通过.传递 tol 或 TRACE 的空矩阵使用默认值.

调用格式六:Q2 = dblquad(FUN,a,b,c,d,[ ],[ ],P1,P2,...)

与 Q2 = dblquad(FUN,a,b,c,d,1.e-6,@QUAD,P1,P2,...) 相同.

说明:(1) 上面所有的被积函数 fun 的调用格式,都需要指定被积函数 fun 为嵌入对象 F = inline('fun'),然后调用 Q2 = dblquad(F,a,b,c,d,...)或指定为函数处理,例如,将被积函数 fun 作一个名为 myfun.m 的 M 文件

```
function z = myfun(x,y)
         z = fun(x,y)
```

然后调用 Q2 = dblquad (@myfun,a,b,c,d,...).

(2) 在定义被积函数 fun 中的运算用数组算子.*, ./和 .^ .

(3) 如果没有给定被积函数 $y=fun$ 的具体表达式,而只给定了积分变量和被积函数的离散的数据或数表,则首先利用插值方法(如分段样条插值,分段低阶的拉格朗日插值等)求出分段插值函数 *fun*,然后用说明(1)中的方法重新指

定插值函数 *fun*.

例 9.8.5 分别用 MATLAB 函数 dblquad 的调用格式二和调用格式三计算 $I=\iint\limits_{D_{xy}}\frac{e^{-(x^2+y^2)}}{x+y}d\sigma$ 的值,取误差限为 $tol=10^{-4}$,并将计算结果与精确值比较.其中 D_{xy}是矩形区域 $0\leqslant x\leqslant 4,1\leqslant y\leqslant 2$.

解 建立并保存被积函数 $f(x,y)=\frac{e^{-(x^2+y^2)}}{x+y}$ 的 M 文件

```
function z = integrnd(x,y)
        z = exp( - (x.^2 + y.^2)) ./(x + y);
```

在 MATLAB 工作窗口输入程序

```
>> a = 0;b = 4;c = 1;d = 2; Q2 = dblquad(@ integrnd,a,b,c,d,1.e - 4)
QL2 = dblquad(@ integrnd,a,b,c,d,1.e - 4,@ quadl)
syms t w
bjh = exp( - (w^2 + t^2)) /(w + t); jfx = int(bjh,w,a,b);
jfy = int(jfx,t,c,d); I2 = double(jfy)
Juewu2 = abs(I2 - Q2), JuewuL2 = abs(I2 - QL2)
```

运行后屏幕显示取误差限为 $tol=10^{-4}$时,分别用 MATLAB 函数 dblquad 的调用格式二和格式三计算 I 的值 Q_2和 QL_2,精确值的近似值 I_2及其它们的绝对误差 $Juewu_2$和 $JuewuL_2$如下

```
Q2  =                        QL2  =
   0.0689                        0.0689
Warning: Explicit integral could not be found.
I2  =               Juewu 2  =              JuewuL2  =
     0.0689              7.8537e - 006            4.5212e - 006
```

由此可见,用 MATLAB 函数 dblquad 的调用格式三比调用格式二计算结果误差小.

例 9.8.6 分别用 MATLAB 函数 dblquad 的调用格式二和调用格式三计算 $I=\iint\limits_{D_{xy}}\frac{\sin(x+y)}{x+y}d\sigma$ 的值,分别取误差限为 $tol=10^{-1},10^{-4}$,并将计算结果与精确值比较.其中 D_{xy}是矩形区域 $0\leqslant x\leqslant 4,-1\leqslant y\leqslant 2$.

解 在 MATLAB 工作窗口输入下列 MATLAB 程序

```
>> a = 0;b = 4;c = - 1;d = 2; Q12 = dblquad(inline(' sin(x + y) ./(x +
y)'),a,b,c,d,1.e - 1)
QL12 = dblquad( inline(' sin(x + y) ./(x + y)'),a,b,c,d,1.e - 1,
@quadl)
Q2 = dblquad(inline(' sin(x + y) ./(x + y)'),a,b,c,d,1.e - 4)
QL2 = dblquad(inline(' sin(x + y) ./(x + y)'),a,b,c,d,1.e - 4,
```

```
@quadl)
 syms t w
bjh = sin(w+t)./(w+t); jfx = int(bjh,w,a,b); jfy = int(jfx,t,
c,d); I2 = double(jfy)
Juewu12 = abs(I2 - Q12), JuewuL12 = abs(I2 - QL12)
Juewu2 = abs(I2 - Q2), JuewuL2 = abs(I2 - QL2)
```

运行后屏幕显示分别取误差限为 $tol = 10^{-1}$ 和 10^{-4} 时,分别用 MATLAB 函数 dblquad 的调用格式二和调用格式三计算 I 的值 Q_{12},Q_2,QL_{12} 和 QL_2,精确值的近似值 I_2 及其它们的绝对误差 $Juewu_{12}$,$Juewu_2$,$JuewuL_{12}$ 和 $JuewuL_2$ 如下

```
Q12 =
  3.64404348383673
QL12 =
  3.64403680642884
Q2 =
   3.64404348383673
QL2 =
  3.64403680642884
I2 =
   3.64403676611252
Juewu12 =
    6.717724212901288e-006
JuewuL12 =
    4.031632050427447e-008
Juewu2 =
    6.717724212901288e-006
JuewuL2 =
    4.031632050427447e-008
```

由此可见,分别取误差限为 $tol = 10^{-1}$ 和 10^{-4} 时,计算结果相同,但是用 MATLAB 函数 dblquad 的调用格式三比格式二计算结果误差小.

9.8.4 一般域上二重积分的数值计算及其 MATLAB 程序

如果二重积分 $\iint_{D_{xy}} f(x,y)\mathrm{d}\sigma$ 的积分区域 D_{xy} 是一般的有界闭区域

$$a \leqslant x \leqslant b, y_1(x) \leqslant y \leqslant y_2(x),$$

其中函数 $y_1(x)$ 和 $y_2(x)$ 皆在区间 $[a,b]$ 上连续,被积函数 $f(x,y)$ 在 D_{xy} 上连续,则可以将二重积分转化为二次积分运算,即

$$I(f)=\iint\limits_{D_{xy}} f(x,y)\,\mathrm{d}\sigma=\int_a^b \mathrm{d}x\int_{y_1(x)}^{y_2(x)} f(x,y)\,\mathrm{d}y. \tag{9.102}$$

数值计算(9.102)的近似值的方法很多,在这里介绍积分区域放大法及其如何用软件 MATLAB 计算.所谓**积分区域放大法**就是根据(9.102)式的积分区域 D_{xy} 作一个矩形区域

$$RT=\{(x,y)\mid a\leqslant x\leqslant b, c\leqslant y\leqslant d, c=\min_{a\leqslant x\leqslant b}\{y_1(x)\}, d=\max_{a\leqslant x\leqslant b}\{y_2(x)\}\},$$

使得 $D_{xy}\subseteq RT$(参见图 9-8).

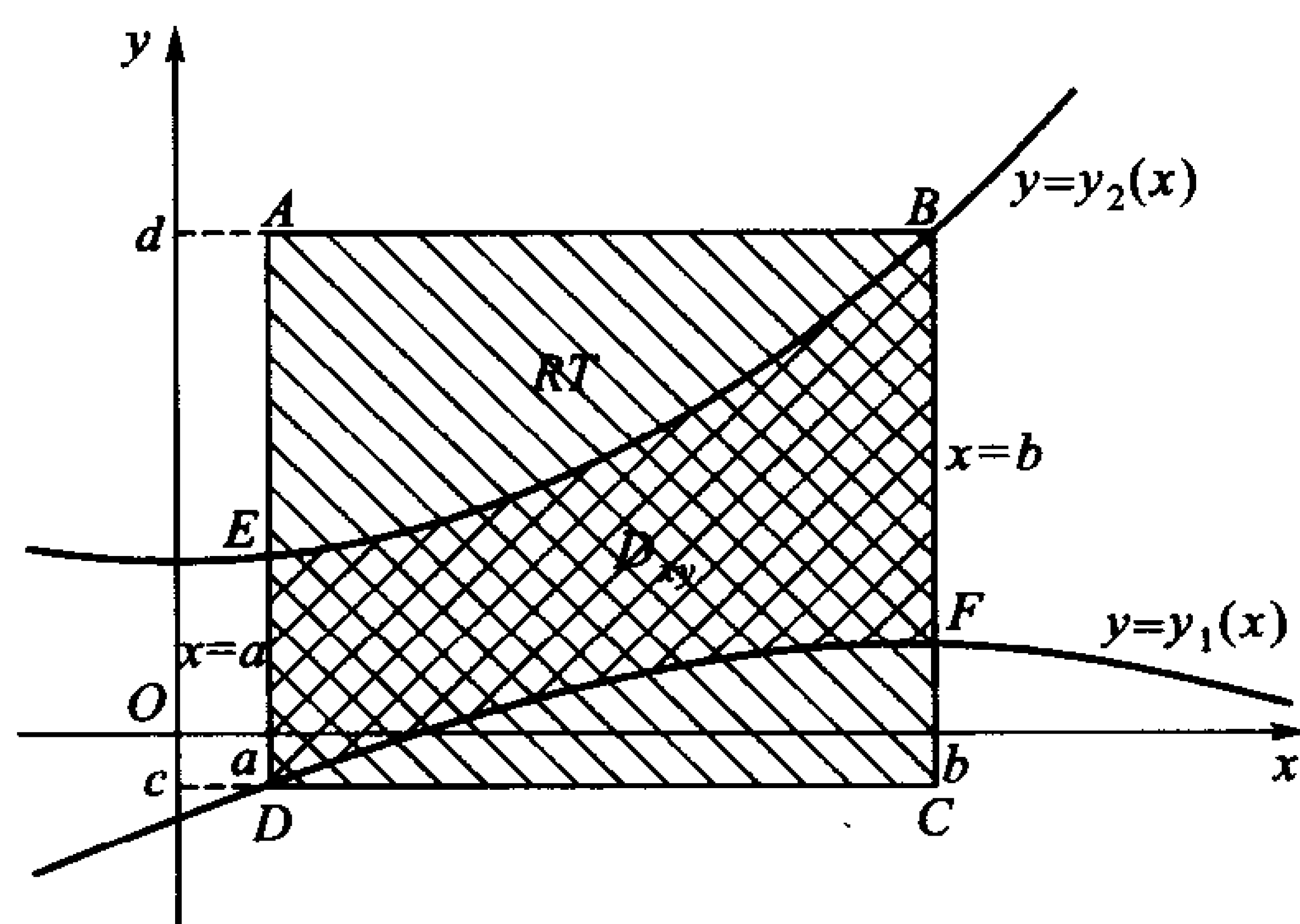


图 9-8 积分区域 D_{xy} 和矩形区域 RT

作辅助函数

$$F(x,y)=\begin{cases} f(x,y), & (x,y)\in D_{xy},\\ 0, & (x,y)\in RT-D_{xy},\end{cases}$$

则根据二重积分复合梯形公式(9.99),得

$$\iint\limits_{D_{xy}} f(x,y)\,\mathrm{d}x\mathrm{d}y=\iint\limits_{RT} F(x,y)\,\mathrm{d}x\mathrm{d}y\cong S_{2n\times 2m}(F(x_k,y_j)),$$

其中 $S_{2n\times 2m}(F(x_k,y_j))$ 是在矩形区域 RT 上的二重积分复合梯形公式(9.99),且满足

$$F(x_k,y_j)=\begin{cases} f(x_k,y_j), & \text{当 } y_1(x_k)\leqslant y_j\leqslant y_2(x),\\ 0, & \text{其他}.\end{cases}$$

例 9.8.7 用 MATLAB 函数 `dblquad` 求直径为 8 的半球的体积 $V_{球}$,误差为 10^{-4}.

解 如图建立空间直角坐标系,则直径为 8 的半球的方程为 $z=\sqrt{4^2-x^2-y^2}$, $(x,y)\in D_{xy}$,其中定义域为 $D_{xy}=\{(x,y)\mid x^2+y^2\leqslant 4^2\}$.则所求的半球的体积 $V_{球}$ 为

$$V_{球} = \iint\limits_{D_{xy}} \sqrt{4^2 - x^2 - y^2}\mathrm{d}x\mathrm{d}y = \int_{-4}^{4} \mathrm{d}x \int_{-\sqrt{4^2-x^2}}^{\sqrt{4^2-x^2}} \sqrt{4^2 - x^2 - y^2}\mathrm{d}y.$$

根据积分区域 D_{xy} 作一个矩形区域

$$RT = \{(x,y) \mid -4 \leqslant x \leqslant 4,\ -4 \leqslant y \leqslant 4\},$$

则 $D_{xy} \subseteq RT$(参见图 9－9).

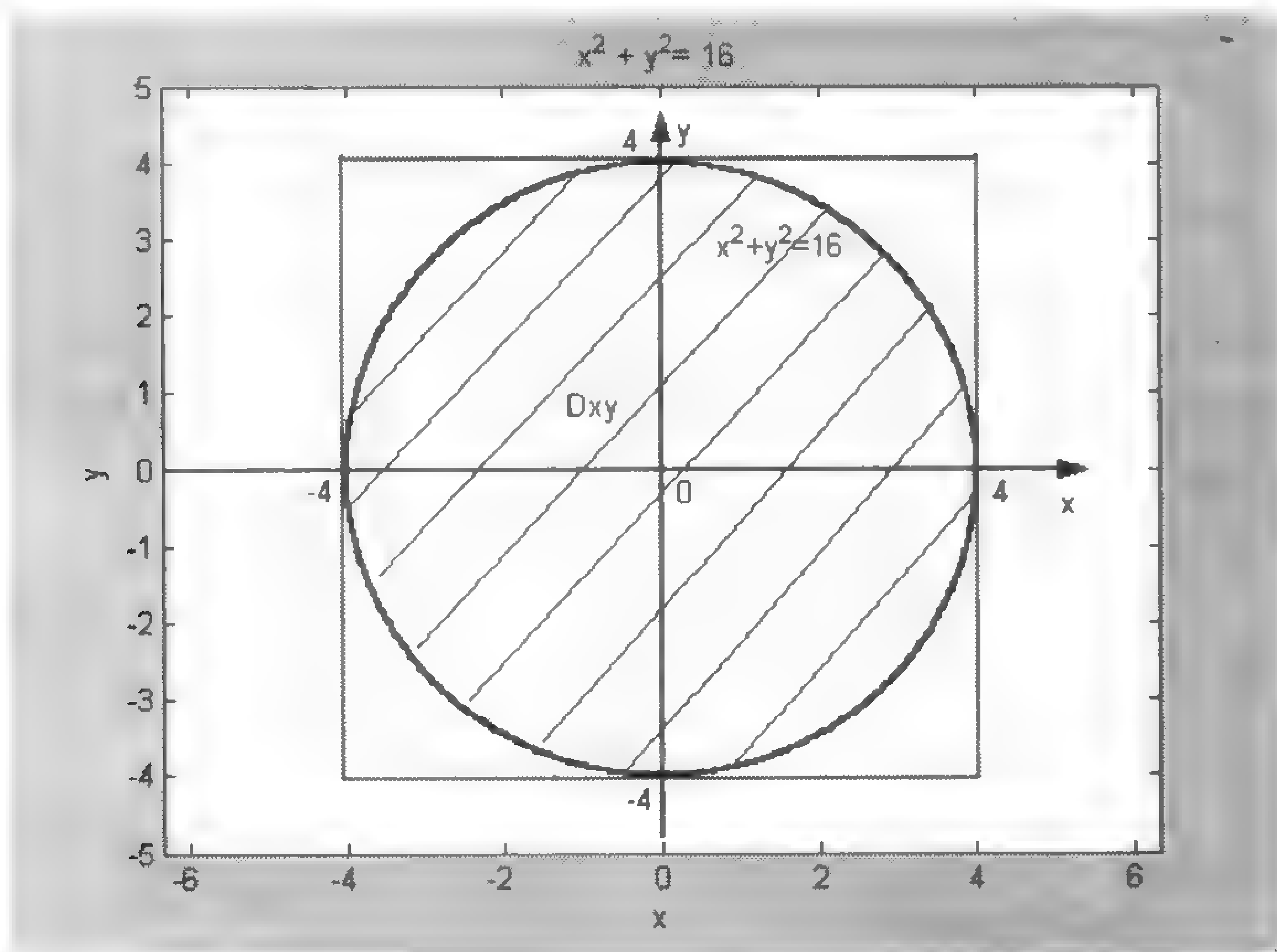


图 9－9　积分区域 D_{xy} 和矩形区域 RT

作辅助函数

$$F(x,y) = \begin{cases} f(x,y), & (x,y) \in D_{xy}, \\ 0, & (x,y) \in RT - D_{xy}, \end{cases}$$

则根据二重积分复合梯形公式(9.99),得

$$\iint\limits_{D_{xy}} \sqrt{4^2 - x^2 - y^2}\mathrm{d}x\mathrm{d}y \cong S_{2n\times 2m}(F(x_k, y_j)),$$

其中 $S_{2n\times 2m}(F(x_k,y_j))$ 是在矩形区域 RT 上的二重积分复合梯形公式(9.99),且满足

$$F(x_k,y_j) = \begin{cases} f(x_k,y_j), & 当 -\sqrt{4^2 - x_k^2} \leqslant y_j \leqslant \sqrt{4^2 - x_k^2}, \\ 0, & 其他. \end{cases}$$

在 MATLAB 工作窗口输入下列 MATLAB 程序

```
>> a = -4;b =4;c = -4;d =4;
V1 =dblquad(inline('sqrt(max(4^2 -(x.^2 +y.^2),0))'),a,b,c,d,
1.e-4,@ quadl)
V =dblquad(inline('sqrt(4^2 -(x.^2 +y.^2)).*(x.^2 +y.^2 < =4^
2)'),a,b,c,d,1.e-4)
syms t w
bjh =sqrt(4^2  (w.^2 +t.^2));
y1 = -sqrt(4^2 -t.^2); y2 =sqrt(4^2 -t.^2);
jfx =int(bjh,w,y1,y2); jfy =int(jfx,t,a,b);
I2 =double(jfy), JuewuL1 =abs(I2 -V1)
Juewu1 =abs(I2 -V), ezplot('x^2 + y^2 - 16',[ -5,5]); axis equal
```

运行后屏幕显示取误差限为 $tol=10^{-4}$ 时,分别用 MATLAB 函数 dblquad 的调用格式二和调用格式三计算 I 的值 V 和 V_1,精确值的近似值 I_2 及其它们的绝对误差 $Juewu_1$ 和 $JuewuL_1$ 如下

```
V1 =                                V =
    1.340434607608882e+002       1.340477821376317e+002
Warning: Explicit integral could not be found.
I2 =                                JuewuL1 =
    1.340412865531645e+002        0.00217420772367
Juewu1 =
   0.00649558446713
```

由此可见,调用函数 quadl 计算二重积分比调用函数 quad 计算的结果误差小,所以,取 134.043 5 作为直径为 8 的半球的体积 $V_{球}$ 的近似值.

9.8.5 三重积分的计算及其 MATLAB 程序

我们可以将数值计算和符号计算二重积分的方法推广到三重积分. 下面分别介绍用 MATLAB 数值计算和符号计算三重积分的方法.

(一) 用 MATLAB 符号计算三重积分

这里主要通过实例介绍将三重积分转化为三次积分运算后,用 MATLAB 进行符号计算的方法. 如果三重积分 $\iiint\limits_V f(x,y,z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z$ 的被积函数 $f(x,y,z)$ 在积分区域 V 上连续,且积分区域为

$$V=\{(x,y,z)\mid a\leqslant x\leqslant b, y_1(x)\leqslant y\leqslant y_2(x), z_1(x,y)\leqslant z\leqslant z_2(x,y)\},$$

则可以将三重积分转化为三次积分运算,即

$$I(f)=\iiint\limits_V f(x,y,z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z=\int_a^b \mathrm{d}x\int_{y_1(x)}^{y_2(x)}\mathrm{d}y\int_{z_1(x,y)}^{z_2(x,y)} f(x,y,z)\,\mathrm{d}z.$$

然后用 MATLAB 函数 int 依次计算三个定积分即可.

例 9.8.8 计算 $I(f)=\iiint\limits_V (x+e^y+\sin z)\mathrm{d}x\mathrm{d}y\mathrm{d}z$,其中积分区域 V 是由旋转抛物面 $z=8-x^2-y^2$,圆柱面 $x^2+y^2=4$ 和 $z=0$ 所围成的空间闭区域.

解 (1) 画出积分区域 V 的草图. 输入程序

```
>> [x,y] = meshgrid( -2:0.01:2);
z1 = 8 - (x.^2 + y.^2);
figure(1)
meshc(x,y,z1)
hold on
x = -2:0.01:2; r = 2;
[x,y,z] = cylinder(r,30)
mesh(x,y,z)
hold off
title('由旋转抛物面 z = 8 - (x^2 + y^2),圆柱面 x^2 + y^2 = 4 和 z = 0 所围成的积分区域 V')
figure(2)
contour(x,y,z,10)
title('由 z = 8 - (x^2 + y^2)和圆柱面 x^2 + y^2 = 4 所围成区域 V 在 x0y 面上的投影区域 Dxy')
```

运行后屏幕显示图 9 - 10.

(2) 确定积分限.输入程序

```
>> syms x y z
f1 = (' z = 8 - (x^2 + y^2)');
f2 = (' x^2 + y^2 = 4 ');
[x,y,z] = solve(f1,f2,x,y,z)
```

运行后屏幕显示旋转抛物面 $z=8-x^2-y^2$ 和圆柱面 $x^2+y^2=4$ 的交线如下

```
x =                         y =             z =
[  (4 - y^2)^(1/2)]        [ y]            [ 4]
[ -(4 - y^2)^(1/2)]        [ y]            [ 4]
```

(3) 输入计算程序

```
>> syms x y z
f = x + exp(y) + sin(z); z1 = 0;z2 = 8 - (x^2 + y^2);
x1 = - sqrt(4 - y^2);x2 = sqrt(4 - y^2);
jfz = int(f,z,z1,z2); jfx = int(jfz,x,x1,x2);
jfy = int(jfx,y, -2,2);jf2 = double(jfy)
```

运行后屏幕显示如下

```
Warning: Explicit integral could not be found.
> In C:\MATLAB6p5p1\toolbox\symbolic\@sym\int.m at line 58
```

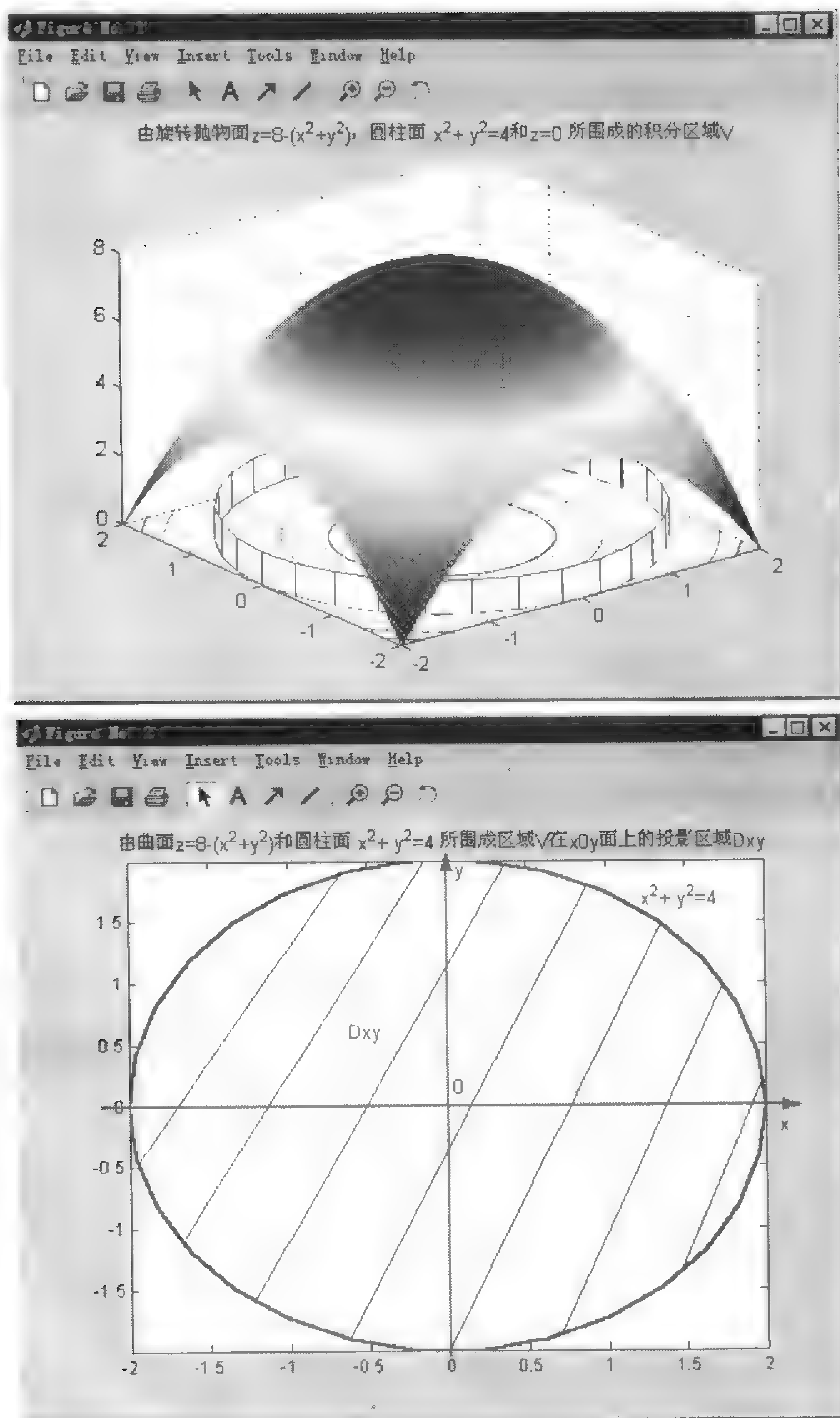


图 9-10 由 $z=8-x^2-y^2$，$x^2+y^2=4$ 和 $z=0$ 围成闭区域 V 及其在 xOy 面上的投影区域 D_{xy}

```
jf2 =
    1.216650998803250e+002
```

因此,所求的 $I(f)$ 的近似值为 121.665 1.

(二) 用 MATLAB 数值计算三重积分

我们可以将数值计算二重积分 $\iint\limits_{D_{xy}} f(x,y)\,\mathrm{d}x\mathrm{d}y$ 的方法推广到三重积分,然后用计算机软件 MATLAB 系统提供的数值计算三重积分 $\iiint\limits_{V} f(x,y,z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z$ 的函数 triplequad.它的调用格式如下:

调用格式一:Q3 = triplequad(FUN,a,b,c,d,p,q)

输入量:*FUN* 是被积函数 $f(x,y,z)$,a,b,c,d,p,q 是三维空间区域 V: $a\leqslant x\leqslant b, c\leqslant y\leqslant d, p\leqslant z\leqslant q$.

输出量:Q_3是用自适应辛普森公式和 MATLAB 函数 quad 计算三重积分 $\iiint\limits_{V} f(x,y,z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z$ 的估计值,其误差限为 10^{-6}.

调用格式二:Q3 = triplequad(FUN,a,b,c,d,p,q,tol)

同上,但指定绝对误差限为 *tol*,默认值为 10^{-6}.

调用格式三:Q3 = triplequad(FUN,a,b,c,d,p,q,tol,@QUADL)

指定用 MATLAB 求积分函数 quadl 代替函数 quad 计算三重积分 $\iiint\limits_{V} f(x,y,z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z$ 的估计值,缺省值是函数 quad.其他同上.

调用格式四:Q3 = triplequad(FUN,a,b,c,d,p,q,tol,@MYQUADF)

指定使用用户自己编写的求三重积分的 MATLAB 程序 MYQUADF.m 代替函数 quad 计算三重积分 $\iiint\limits_{V} f(x,y,z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z$ 的估计值,其中程序 MYQUADF.m 的调用顺序应该与 quadl 和 quad 的相同,缺省值是函数 quad.其他同上.

调用格式五:Q3 = triplequad(FUN,a,b,c,d,p,q,tol,@QUADL,P1,P2,...)

此命令规定对函数 fun (x,y,z,P1,P2,...)附加的参数 $P_1,P_2,\cdots$ 直接通过. 传递 tol 或 TRACE 的空矩阵使用默认值.

调用格式六:Q3 = triplequad(FUN,a,b,c,d,p,q,[],[],P1,P2,...)

与 Q3 = triplequad(FUN,a,b,c,d,p,q,1.e-6,@QUAD,P1,P2,...)相同.

说明:(1) 上面所有的被积函数 *fun* 的调用格式,都必须指定被积函数 *fun* 为嵌入对象 F = inline('fun'),然后调用 Q3 = triplequad (F,a,b,...)或指定为函数处理,例如,将被积函数 *fun* 作一个名为 myfun.m 的 M 文件:

```
function u = myfun(x,y,z)
         u = fun(x,y,z)
```

然后调用 Q3 = triplequad (@myfun,a,b,c,d,p,q,...).

(2) 在定义被积函数 *fun* 中的运算用数组算子. * , ./ 和 .^ ;

(3) 如果没有给定被积函数 $y = fun$ 的具体表达式,而只给出了积分变量和被积函数的离散的数据或数表,则首先利用插值方法(如分段样条插值,分段低阶的拉格朗日插值等)求出分段插值函数 *fun*,然后用说明(1)中的方法重新指定插值函数 *fun*.

例 9.8.9 分别用 MATLAB 函数 triplequad 的调用格式二和调用格式三计算 $I(f) = \iiint\limits_V (x + e^y + \sin z)\mathrm{d}x\mathrm{d}y\mathrm{d}z$ 的值,取误差限为 $tol = 10^{-4}$,并将计算结果与精确值比较. 其中 V 是三维长方体区域 $-2 \leqslant x \leqslant 2$, $-2 \leqslant y \leqslant 2, 0 \leqslant z \leqslant 4$.

解 建立并保存被积函数 $f(x,y,z) = x + e^y + \sin z$ 的 M 文件

```
function u = integrnd1(x,y,z)
         u = x + exp(y) + sin(z);
```

在 MATLAB 工作窗口输入程序

```
>>  a = -2;b = 2;c = -2;d = 2;p = 0;q = 4;
Q3 = triplequad(@integrnd1,a,b,c,d,p,q,1.e-4)
QL3 = triplequad(@integrnd1,a,b,c,d,p,q,1.e-4,@quadl)
  syms x y z
f = x + exp(y) + sin(z);
 jfz = int(f,z,p,q); jfy = int(jfz,y,c,d);
jfx = int(jfy,x,a,b); I3 = double(jfx),
Juewu3 = abs(I3 - Q3)
JuewuL3 = abs(I3 - QL3)
```

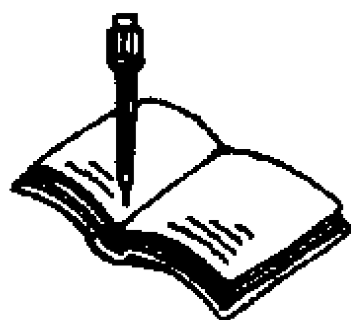
运行后屏幕显示取误差限为 $tol = 10^{-4}$ 时,分别用 triplequad 的调用格式二和调用格式三计算 I 的值 Q_3 和 QL_3,精确值的近似值 I_3 及其它们的绝对误差 $Juewu_3$ 和 $JuewuL_3$ 如下

```
Q3 =
      1.425178451284647e+002,
QL3 =
      1.425178451284647e+002
I3 =
      1.425178309849224e+002,
Juewu3 =
      1.414354233020276e-005
JuewuL3 =
```

```
        1.414354233020276e-005
```

用 MATLAB 函数 `triplequad` 的调用格式三与格式二计算三重积分 I 的值都等于 142.517 8，与精确值 I_3 的绝对误差为 1.414 4e-005，近似于要求的误差限 $tol=10^{-4}$.

依照上面介绍的二重积分和三重积分的数值计算和符号计算的方法，读者不难将这些方法推广到更多重的积分上去.



习 题 9.8

1. 计算 $\iint\limits_{D_{xy}} e^{-(x^2+y^2)}\mathrm{d}\sigma$，其中 D_{xy} 是曲线 $xy=1, y=\sqrt{x}, x=2$ 所围成的平面区域.

2. 计算 $\iint\limits_{D_{xy}} \dfrac{\sin(x^2+y^2)}{x+y}\mathrm{d}\sigma$，其中 D_{xy} 是由曲线 $x=y^2, y=x-2$ 所围成的平面区域.

3. 根据(9.88)式编写 MATLAB 程序，计算 $I=\iint\limits_{D_{xy}} \dfrac{\cos(x+y)}{x+y}\mathrm{d}\sigma$ 的值，并将计算结果与精确值比较. 其中 D_{xy} 是矩形区域 $0\leqslant x\leqslant 4, -1\leqslant y\leqslant 2$.

4. 根据(9.93)式编写 MATLAB 程序计算 $I=\iint\limits_{D_{xy}} \dfrac{e^{-(x^2+y^2)}}{2x+y^3}\mathrm{d}\sigma$ 的值，分别取 $(n,m)=(2,2),(12,12),(112,112),(1\,112,1\,112)$，并将计算结果与精确值比较. 观察节点的个数与误差的关系. 问取 $(n,m)=(111\,112,111\,112)$ 时，计算机能计算吗，为什么？其中 D_{xy} 是矩形区域 $0\leqslant x\leqslant 4, 1\leqslant y\leqslant 3$.

5. 分别用 MATLAB 函数 `dblquad` 的调用格式一和调用格式二计算 $I=\iint\limits_{D_{xy}} \dfrac{e^{-(x^2+y^2)}}{5x^2+3y}\mathrm{d}\sigma$ 的值，取误差限为 $tol=10^{-4}$，并将计算结果与精确值比较. 其中 D_{xy} 是矩形区域 $0\leqslant x\leqslant 4, 1\leqslant y\leqslant 2$.

6. 分别用 MATLAB 函数 `dblquad` 的调用格式二和调用格式三计算 $I=\iint\limits_{D_{xy}} \dfrac{\sin(7x^2+y)}{x+y}\mathrm{d}\sigma$ 的值. 分别取误差限为 $tol=10^{-1}, 10^{-4}$，并将计算结果与精确值比较. 其中 D_{xy} 是矩形区域 $0\leqslant x\leqslant 4, -1\leqslant y\leqslant 2$.

7. 用 MATLAB 函数 `dblquad` 求椭球 $\dfrac{x^2}{4}+\dfrac{y^2}{16}+\dfrac{z^2}{9}=1$ 的体积 $V_{球}$，误差为 10^{-4}.

8. 计算 $I(f)=\iiint\limits_{V}(5x^3+e^{x+y}+\cos z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z$，其中积分区域 V 是由旋转抛物面 $z=x^2+y^2$，圆柱面 $x^2+y^2=4$ 和 $z=0$ 所围成的空间闭区域.

9. 分别用 MATLAB 函数 `triplequad` 的调用格式二和调用格式三计算 $I(f)=\iiint\limits_{V}(x+e^{y}-\cos z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z$ 的值，取误差限为 $tol=10^{-4}$，并将计算结果与精确值比较. 其中 V 是

三维长方体区域 $-2\leqslant x\leqslant 2,-2\leqslant y\leqslant 2,0\leqslant z\leqslant 4$.

10. 根据函数 $z=f(x,y)$ 的数据表

z \ x / y	0	0.5	1	1.5	2
1	10	12	13	15	14
1.2	14	15	17	18	16
1.4	15	16	18	17	15
1.6	13	14	16	15	13

使用复合梯形公式计算二重积分 $\iint\limits_{D_{xy}} f(x,y)\,\mathrm{d}x\mathrm{d}y$,其中积分区域 D_{xy} 是 $0\leqslant x\leqslant 2,1\leqslant y\leqslant 1.6$.

11. 根据步长 $h_x=h_y=0.5$ 的复合辛普森公式计算二重积分 $\iint\limits_{D_{xy}} \mathrm{e}^{\sin x}\mathrm{d}x\mathrm{d}y$,其中积分区域 D_{xy} 是 $0\leqslant x\leqslant 3,0\leqslant y\leqslant x$.

第十章 常微分方程(组)求解

微分方程在科技、工程、经济管理以及生态、环境、人口、交通等各个领域中常用于建立数学模型,它是研究函数变化规律的有力工具.如在研究弹性物体的振动,电阻、电容、电感电路的瞬变、热量在介质中的传播、抛射体的轨迹以及污染物浓度的变化、人口增长的预测、种群数量的演变、交通流量的控制等等过程中,作为研究对象的函数,要和函数的导数一起,用一个符合其内在规律的方程,即微分方程来描述.

建立微分方程只是解决问题的第一步,通常需要求出方程的解来说明实际现象,并加以检验.如果能得到解析解固然是便于分析和应用的,但是我们知道,只有线性常系数微分方程,并且自由项是某些特殊类型的函数时,才可以肯定得到这样的解,而绝大多数变系数方程、非线性方程都很难得到解析解.另外,有时即使能求出解析解,也会由于很难从解析解中计算函数 $y(x)$ 的值而不实用.例如,容易求出初值问题

$$\begin{cases} \dfrac{dy}{dx} = 1 - y\cos x, 0 \leqslant x \leqslant T, \\ y(0) = 0, \end{cases}$$

的解为

$$y(x) = e^{-\sin x}\int_0^x e^{\sin t}dt.$$

但是,对于给定的 $x = x_0$,要计算函数值 $y(x_0)$ 还需要用数值积分的方法.于是对于用微分方程解决的实际问题来说,数值解法就是一个十分重要的手段.为了将微分方程的数值解与解析解比较,本章首先简单介绍用 MATLAB 解微分方程的解析解(符号解)的方法,然后重点介绍微分方程的数值解法及其 MATLAB 程序.

10.1 常微分方程(组)的 MATLAB 符号求解

在 MATLAB 系统中提供了常微分方程(组)符号解(symbolic solution of ordinary differential equations)的函数 `dsolve`.在调用此函数前,必须首先将给定的常微分方程(组)中的一阶导数用 D 表示,例如,$\frac{dy}{dx}$写成 Dy. n 阶导数用 Dn 表

示,例如,$\frac{d^6y}{dx^6}$表示为 $D6y$. 由此,常微分方程 $y''+4y'=y^2+x^2$ 写成 $D2y+4Dy=y^2+x^2$. 下面具体介绍函数 dsolve 解常微分方程(组)的方法.

10.1.1 用 MATLAB 求常微分方程(组)的通解

用 MATLAB 函数 dsolve 求常微分方程

$$F(x,y,y',y'',\cdots,y^{(n)})=0 \tag{10.1}$$

的通解的主要调用格式如下:

调用格式一: S = dsolve ('eqn','var')

输入的量:eqn 是常微分方程(10.1)改用符号方程表示的常微分方程 $F(x,y,Dy,D2y,\cdots,Dny)=0$. Var 表示自变量,默认的自变量为 t.

输出的量:S 是常微分方程(10.1)的通解.

求常微分方程组

$$\begin{cases} F_1(x,y_1,y_1',y_1'',\cdots,y_1^{(n_1)})=0, \\ F_2(x,y_2,y_2',y_2'',\cdots,y_2^{(n_2)})=0, \\ \cdots\cdots\cdots\cdots \\ F_m(x,y_m,y_m',y_m'',\cdots,y_m^{(n_m)})=0 \end{cases} \tag{10.2}$$

的通解的主要调用格式如下:

调用格式二: S = dsolve ('eqn1','eqn2',...,'eqnm','var')

输入的量:eqn1, eqn2 ,... , eqnm 分别是常微分方程组(10.2)中用符号方程表示的 m 个常微分方程. 默认的自变量为 t,也可以将自变量 t 变为其他的符号变量 var.

输出的量:S 是常微分方程组(10.2)的通解.

例 10.1.1 求下列常微分方程的通解:

(1) $\frac{dy}{dt}=-at$;

(2) $\frac{d^2y}{dx^2}+a\frac{dy}{dx}=b(\sin x+\cos x)$;

(3) $\frac{dy}{dx}=y^2(1-y)$;

(4) $\frac{d^2x}{dt^2}+\frac{a}{m}\frac{dx}{dt}+\frac{b}{m}x=\frac{F(t)}{m}$;

(5) $\frac{d^2y}{dx^2}=\sin y$.

解 输入程序

```
>>y1 =dsolve('Dy = -a*t')
y2 =dsolve('D2y +a*Dy =b*(sin(x) +cos(x))','x')
y3 = dsolve('Dy  = y^2*(1 -y)'),
x4 -dsolve('D2x +a*Dx/m +b*x/m - F(t)/m'),
```

```
y5 = dsolve('D2y = sin(y)'), pretty(y5)
```

运行后屏幕显示常微分方程(1),(2),(3),(4)和(5)的通解 y_1,y_2,y_3,x_4和 y_5依次如下

```
y1 =
C1 * exp( - a * t)
y2 =
-b * (a * cos(x) - a * sin(x) + sin(x) + cos(x))/(a^2 +1) + C1 +
C2 * exp( - a * x)
Warning: Explicit solution could not be found; implicit solu-
tion returned.
> In C:\MATLAB6p5\toolbox\symbolic\dsolve.m at line 292
y3 =
t +1/y - log(y) + log( -1 +y) + C1 =0
x4 =
( - int(F(t) * exp(1/2 * (a + (a^2 - 4 * m * b)^(1/2)) * t/m),t) *
exp(1/2 * (a - (a^2 -4 * m * b)^(1/2)) * t/m) + int(F(t) * exp(1/2 * (a - (a^
2 -4 * m * b)^(1/2)) * t/m),t) * exp(1/2 * (a + (a^2 -4 * m * b)^(1/2)) * t/
m)) * exp( -a * t/m)/(a^2 -4 * m * b)^(1/2) + C1 * exp( -1/2 * (a + (a^2 -4 *
m * b)^(1/2)) * t/m) + C2 * exp( -1/2 * (a - (a^2 -4 * m * b)^(1/2)) * t/m)
Warning: Explicit solution could not be found; implicit solu-
tion returned.
> In C:\MATLAB6p5\toolbox\symbolic\dsolve.m at line 292
y5 =
[Int(1/( -2 * cos(a) + C1)^(1/2),a - "..y) - t - C2 - 0,
 - Int(1/( -2 * cos(a) + C1)^(1/2),a = "..y) - t - C2 =0]
```

其中 C_1和 C_2是任意常数. 因为常微分方程(3)和(5)的显式解没有被找到,所以返回隐式解.

例 10.1.2 求下列常微分方程组的通解:

$$(1)\begin{cases}\dfrac{\mathrm{d}f(t)}{\mathrm{d}t}=af(t)+bg(t),\\ \dfrac{\mathrm{d}g(t)}{\mathrm{d}t}=-af(t)+bg(t);\end{cases}\qquad (2)\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}x}+4z=\sin x,\\ \dfrac{\mathrm{d}y}{\mathrm{d}x}+3\dfrac{\mathrm{d}z}{\mathrm{d}x}=\cos x.\end{cases}$$

解 输入程序

```
>> S1 = dsolve('Df = a * f + b * g','Dg = -a * f + b * g'),
f = S1.f,g = S1.g,de1 = 'Dy + 4 * z = sin(x)';
de2 = 'Dy + 3 * Dz = cos(x)';
S2 = dsolve(de1, de2, 'x'), Y = S2.y,z = S2.z
```

运行后屏幕显示微分方程组(1)和(2)的通解

```
S1 =
     f: [1x1 sym]
     g: [1x1 sym]
f =
C1*exp(1/2*a*t+1/2*t*b-1/2*t*(a^2-6*a*b+b^2)^(1/2))+C2*exp(1/2*a*t+1/2*t*b+1/2*t*(a^2-6*a*b+b^2)^(1/2))
g =
-1/2*(C1*a*exp(1/2*a*t+1/2*t*b-1/2*t*(a^2-6*a*b+b^2)^(1/2))-C1*exp(1/2*a*t+1/2*t*b-1/2*t*(a^2-6*a*b+b^2)^(1/2))*b+C1*(a^2-6*a*b+b^2)^(1/2)*exp(1/2*a*t+1/2*t*b-1/2*t*(a^2-6*a*b+b^2)^(1/2))+C2*a*exp(1/2*a*t+1/2*t*b+1/2*t*(a^2-6*a*b+b^2)^(1/2))-C2*exp(1/2*a*t+1/2*t*b+1/2*t*(a^2-6*a*b+b^2)^(1/2))*b-C2*(a^2-6*a*b+b^2)^(1/2)*exp(1/2*a*t+1/2*t*b+1/2*t*(a^2-6*a*b+b^2)^(1/2)))/b
S2 =
     y: [1x1 sym]
     z: [1x1 sym]
Y =
C1-3*exp(4/3*x)*C2+3*C2+4/25*sin(x)+3/25*cos(x)
z =
exp(4/3*x)*C2-1/25*cos(x)+7/25*sin(x)
```

10.1.2 用 MATLAB 求常微分方程(组)的特解

如果给定常微分方程(10.1)的初始条件

$$y(x_0)=a_0,y'(x_0)=a_1,\cdots,y^{(n)}(x_0)=a_n, \tag{10.3}$$

则求方程(10.1)的特解的主要调用格式如下：

调用格式三： S = dsolve ('eqn','condition1',...,'conditionn','var')

输入的量：eqn 是常微分方程(10.1)改用符号方程表示的常微分方程 $F(x,y,Dy,D2y,...,Dny)=0$；condition1,...,conditionn 是初始条件(10.3)；var 表示自变量，默认的自变量为 t.

输出的量：S 是常微分方程(10.1)的特解.

同理，如果给定常微分方程组(10.2)的初始条件，则求方程(10.2)的特解的主要调用格式如下：

调用格式四： S = dsolve('eqn1','eqn2',...,'eqnm','condition1','condition2',...,'var')

输入的量：eqn1, eqn2, ...,eqnm 分别是常微分方程组(10.2)中用符号方

程表示的 m 个常微分方程;condition1,condition2,...是常微分方程组(10.2)的初始条件;默认的自变量为 t,也可以将自变量 t 变为其他的符号变量 var.

输出的量:S 是常微分方程组(10.2)的特解.

例 10.1.3 求下列常微分方程在给定初始条件下的特解:

(1) $\left(\dfrac{\mathrm{d}y}{\mathrm{d}x}\right)^2+y^2=1, y(0)=0$;

(2) $\dfrac{\mathrm{d}^3w}{\mathrm{d}t^3}=-w, w(0)=1, \left.\dfrac{\mathrm{d}w}{\mathrm{d}t}\right|_{t=0}=0, \left.\dfrac{\mathrm{d}^2w}{\mathrm{d}t^2}\right|_{t=0}=0$;

(3) $\dfrac{\mathrm{d}^2f(x)}{\mathrm{d}x^2}=-a^2\dfrac{\mathrm{d}f(x)}{\mathrm{d}x}, f(0)=1, \left.\dfrac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=\frac{\pi}{a}}=0.$

解 输入程序

```
>> y = dsolve('(Dy)^2 + y^2 = 1','y(0) = 0')
w = dsolve('D3w = -w','w(0)=1, Dw(0)=0, D2w(0)=0')
f = dsolve('D2f = -a^2*Df', 'f(0) = 1, Df(pi/a) = 0')
```

运行后屏幕显示常微分方程(1),(2)和(3)在给定初始条件下的特解 y,w,f 依次如下

```
y =
  [  sin(t)]
  [ -sin(t)]
w =
  1/3*exp(-t)+2/3*exp(1/2*t)*cos(1/2*3^(1/2)*t)
f =
  1
```

例 10.1.4 求下列微分方程组满足所给初始条件的特解:

(1) $\begin{cases}\dfrac{\mathrm{d}f(t)}{\mathrm{d}t}=af(t)+bg(t),\\[2mm] \dfrac{\mathrm{d}g(t)}{\mathrm{d}t}=-af(t)+bg(t),\end{cases} \quad f(0)=1, g(0)=2$;

(2) $\begin{cases}\dfrac{\mathrm{d}^2y}{\mathrm{d}x^2}+2\dfrac{\mathrm{d}y}{\mathrm{d}x}+y+\dfrac{\mathrm{d}z}{\mathrm{d}x}+z=0,\\[2mm] \dfrac{\mathrm{d}y}{\mathrm{d}x}+y+\dfrac{\mathrm{d}^2z}{\mathrm{d}x^2}+2\dfrac{\mathrm{d}z}{\mathrm{d}x}+z=\mathrm{e}^x,\end{cases} \quad y(0)=0, \left.\dfrac{\mathrm{d}y}{\mathrm{d}x}\right|_{x=3}=2, z(5)=1, \left.\dfrac{\mathrm{d}z}{\mathrm{d}x}\right|_{x=0}=0.$

解 输入程序

```
>> S1 = dsolve('Df = a*f + b*g','Dg = -a*f + b*g','f(0) = 1',
'g(0) = 2')
f = S1.f; fs = simple(f), g = S1.g;
gs = simple(g), de1 = 'D2y + 2*Dy + y + Dz + z = 0';
```

```
de2 ='Dy+y+D2z+2*Dz+z=exp(x)';
S2 =dsolve(de1,de2,'y(0)=0, Dy(3)=2, z(5)=1, Dz(0)=0')
y =S2.y; ys =simple(y), z =S2.z; zs =simple(z),
```

运行后屏幕显示常微分方程组(1)和(2)在给定初始条件下的特解 *fs*,*gs*,*ys* 和 *zs* 依次如下

```
S1 =
     f: [1x1 sym]
     g: [1x1 sym]
fs =
-1/2*(a+3*b-(a^2-6*a*b+b^2)^(1/2))/(a^2-6*a*b+b^2)^(1/2)*exp(1/2*a*t+1/2*t*b 1/2*t*(a^2-6*a*b+b^2)^(1/2))+1/2*(a+3*b+(a^2-6*a*b+b^2)^(1/2))/(a^2-6*a*b+b^2)^(1/2)*exp(1/2*a*t+1/2*t*b+1/2*t*(a^2-6*a*b+b^2)^(1/2))
gs =
(2*exp(1/2*(a+b-(a^2-6*a*b+b^2)^(1/2))*t)*a-exp(1/2*(a+b-(a^2-6*a*b+b^2)^(1/2))*t)*b+exp(1/2*(a+b-(a^2-6*a*b+b^2)^(1/2))*t)*(a^2-6*a*b+b^2)^(1/2)-2*exp(1/2*(a+b+(a^2-6*a*b+b^2)^(1/2))*t)*a+exp(1/2*(a+b+(a^2-6*a*b+b^2)^(1/2))*t)*b+exp(1/2*(a+b+(a^2-6*a*b+b^2)^(1/2))*t)*(a^2-6*a*b+b^2)^(1/2))/(a^2-6*a*b+b^2)^(1/2)
S2 =
     y: [1x1 sym]
     z: [1x1 sym]
ys =
-1/2*exp(-3)*(-2*exp(x-2)-exp(-1-t+x)+exp(1+x)-4*exp(-1 t)+exp( 1+x)+4*exp(-1)+2*exp(4-t+x)-2*exp(x+4)+8*exp(4-t)-8*exp(4)+2*exp(-2*t+6)+4*exp(-2*t+9)+2*exp(1+x)*t+2*exp(x+3)*t-exp(-4+x)*t-12*exp(x+3)-2*exp(6)-6*exp(x-2*t+6)+2*exp(-2-t+x)-exp(x-t+9)+12*exp(x-t+3)+4*exp(3)+6*exp(x+6)-exp(x-2*t+1)-exp(x+6)*t-4*exp(3-t)-4*exp(9-t)+exp(x-2*t+9))/(2*exp(-2)+2-exp(-7)-exp(3))
zs =
-1/2*(4+3*exp(x-2)+4*exp(-4)-2*exp(1+x)-8*exp(1)-exp(x-7)+8*exp(6-t)-4*exp(-2*t+6)+2*exp(x-t+6)+exp(-4+x)+exp(x-2*t-2)-exp(x-7)*t+5*exp(x+3)+2*exp(x-2)*t-exp(x-t-7)-10*exp(x)-exp(x-2*t+6)-13*exp(x-t+3)-exp(x+3)*t-2*exp(3)+2*exp(x-t)+6*exp(x-2*t+3)-2*exp(x-5)+2*exp(x)*t-2*exp(-2*t+3)+4*exp(3-t))/(-2*exp(-2)-2+exp(-7)+
```

```
exp(3))
```

10.1.3 线性常微分方程组的 MATLAB 解法

如果线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}+\boldsymbol{B}$ 中的未知量较多,用上面的求解方法就不方便,这时我们可以用下面介绍的求解齐次线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}$ 和非齐次线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}+\boldsymbol{B}(\boldsymbol{B}\neq\boldsymbol{0})$ 的方法求解.

求解含未知量较多的齐次线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}$ 可以用下面的定理.

定理 10.1 如果齐次线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}$ 的 n 阶矩阵 $\boldsymbol{A}$ 有 n 个线性无关的特征向量 $\boldsymbol{\xi}_1,\boldsymbol{\xi}_2,\cdots,\boldsymbol{\xi}_n$,它们分别对应的特征值依次为 $\lambda_1,\lambda_2,\cdots,\lambda_n$,则 $\boldsymbol{Y}=\boldsymbol{AX}$ 的一个基础解系矩阵为

$$\boldsymbol{X}(t)=(\mathrm{e}^{\lambda_1 x},\mathrm{e}^{\lambda_2 x},\cdots,\mathrm{e}^{\lambda_n x}),\ -\infty<x<+\infty. \tag{10.4}$$

根据定理 10.1,编写求解齐次线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}$ 的 MATLAB 程序如下:

求解齐次线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}$ 的 MATLAB 主程序
输入量:齐次线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}$ 的矩阵 $\boldsymbol{A}$.
输出量:$\boldsymbol{X}$ 是 $\boldsymbol{Y}=\boldsymbol{AX}$ 的解,$\boldsymbol{E}$ 是矩阵 $\boldsymbol{A}$ 的特征值,$\boldsymbol{V}$ 是矩阵 $\boldsymbol{A}$ 的特征向量.

名为 qcxxcwz.m 的求解齐次线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}$ 的 MATLAB 主程序如下:

```
function [X,E,V] = qcxxcwz(A)
syms x
E = eig(A); [V,n] = eig(A);
X = exp(E * x)' * V;
```

例 10.1.5 用两种方法求下面常微分方程组的通解:

$$\begin{cases}\dfrac{\mathrm{d}f(t)}{\mathrm{d}t}=2f(t)+3g(t),\\[2mm]\dfrac{\mathrm{d}g(t)}{\mathrm{d}t}=-2f(t)+3g(t).\end{cases}$$

解 输入程序

```
>>syms C1 C2
A = [2,3; -2,3]; [X,E,V] = qcxxcwz(A), S = X * [C1;C2]
S1 = dsolve('Df = 2 * f  +3 * g','Dg = -2 * f +3 * g','x'),f = S1.f,g =
S1.g
```

运行后屏幕显示用两种方法求上面常微分方程组的结果依次如下

```
X =
[ 1/5 * exp((5/2 -1/2 * i * 23^(1/2)) * conj(x)) * 15^(1/2) +
exp((5/2 +1/2 * i * 23^(1/2)) * conj(x)) * (1/30 * 15^(1/2) +1/30 * i *
```

```
345^(1/2)),
1/5*exp((5/2-1/2*i*23^(1/2))*conj(x))*15^(1/2)+exp((5/2+
1/2*i*23^(1/2))*conj(x))*(1/30*15^(1/2)-1/30*i*345^(1/2))]
        E =
           2.5000 + 2.3979i
           2.5000 - 2.3979i
        V =
           0.7746              0.7746
           0.1291 + 0.6191i   0.1291 - 0.6191i
        S =
        (1/5*exp((5/2-1/2*i*23^(1/2))*conj(x))*15^(1/2)+exp
((5/2+1/2*i*23^(1/2))*conj(x))*(1/30*15^(1/2)+1/30*i*345^
(1/2)))*C1+(1/5*exp((5/2-1/2*i*23^(1/2))*conj(x))*15^(1/2)+
exp((5/2+1/2*i*23^(1/2))*conj(x))*(1/30*15^(1/2)-1/30*i*345
^(1/2)))*C2
        S1 =
             f: [1x1 sym]
             g: [1x1 sym]
        f =
        1/23*exp(5/2*x)*(23*C1*cos(1/2*x*23^(1/2))-23^(1/2)
*sin(1/2*x*23^(1/2))*C1+6*23^(1/2)*sin(1/2*x*23^(1/2))*C2)
        g =
         -1/23*exp(5/2*x)*(4*23^(1/2)*sin(1/2*x*23^(1/2))*
C1-23*C2*cos(1/2*x*23^(1/2))-23^(1/2)*sin(1/2*x*23^(1/2))*
C2)
```

由于求解非齐次线性常微分方程组 $\boldsymbol{Y}=\boldsymbol{AX}+\boldsymbol{B}(\boldsymbol{B}\neq\boldsymbol{0})$ 的方法只比上面的方法多了一个积分过程，所以留给读者完成.



习 题 10.1

1. 求下列常微分方程的通解：

(1) $\dfrac{d^6y}{dt^6}=-at^7+\cos 5t$；　　(2) $\dfrac{d^2y}{dx^2}+5\dfrac{dy}{dx}=7x(\sin 3x+\cos 3x)$；

(3) $\dfrac{d^2x}{dt^2}+\beta\dfrac{dx}{dt}+\gamma x=\dfrac{F(t)}{m}$；　　(4) $\dfrac{d^2y}{dx^2}+2\dfrac{dy}{dx}+y=\sin x$.

2. 求下列常微分方程组的通解：

(1) $\begin{cases} \dfrac{df(t)}{dx} = 5f(t) + 7g(t), \\ \dfrac{dg(t)}{dx} = -5f(t) + 7g(t); \end{cases}$　　(2) $\begin{cases} 3\dfrac{dy}{dx} + 8z = \sin 5x, \\ \dfrac{dy}{dx} + 3\dfrac{dz}{dx} = \cos 5x. \end{cases}$

3. 求下列常微分方程在给定初始条件下的特解：

(1) $\left(\dfrac{dy}{dx}\right)^2 + y^2 = \cos x, y(0) = 0;$

(2) $\dfrac{d^3 w}{dt^3} = -w, w(0) = 1, \left.\dfrac{dw}{dt}\right|_{t=0} = 1, \left.\dfrac{d^2 w}{dt^2}\right|_{t=0} = 0;$

(3) $\dfrac{d^2 f(x)}{dx^2} = 4\dfrac{df(x)}{dx}, f(0) = 1, \left.\dfrac{df(x)}{dx}\right|_{x=\frac{\pi}{2}} = 0.$

4. 求下列微分方程组满足所给初始条件的特解：

(1) $\begin{cases} \dfrac{df(t)}{dt} = 2f(t) + 3g(t), \\ \dfrac{dg(t)}{dt} = -2f(t) + 3g(t), \end{cases} f(0) = 1, g(0) = 2;$

(2) $\begin{cases} \dfrac{d^2 y}{dx^2} + 2\dfrac{dy}{dx} + y + \dfrac{dz}{dx} + z = 0, \\ \dfrac{dy}{dx} + y + \dfrac{d^2 z}{dx^2} + 2\dfrac{dz}{dx} + z = e^x, \end{cases} y(1) = 0, \left.\dfrac{dy}{dx}\right|_{x=3} = 2, z(5) = 1, \left.\dfrac{dz}{dx}\right|_{x=2} = 3.$

5. 用两种方法求下面常微分方程组的通解：

$$\begin{cases} \dfrac{df(t)}{dt} = 5f(t) + 7g(t), \\ \dfrac{dg(t)}{dt} = -5f(t) + 7g(t). \end{cases}$$

10.2　求初值问题的数值解的基本思想

在上节的讨论中我们看到，利用符号运算可以寻求常微分方程(组)的初等函数公式解，或称符号解(包含通解和特解). 但是存在有这种解的常微分方程(组)的类型很少，往往只局限于线性常系数常微分方程(组)或少数低阶的常微分方程(组)以及少数线性变系数常微分方程(组). 对于更加广泛的、非线性的一般常微分方程(组)，通常不存在有初等函数公式解. 由于实际问题求解的需要，经常采用数值解法求常微分方程(组)的数值解. 常见的数值解法有欧拉(Euler)方法、亚当斯(Adams)方法、龙格－库塔(Runge-Kutta)方法等. 因为龙格－库塔方法的精度较高，计算量适中，常被人们采用.

解微分方程通常需要附加某种定解条件. 微分方程和定解条件组成定解问题. 定解条件通常有两种方法给出：一种是给出积分曲线在初始时刻的性态，这类条件称为初始条件，对应的定解问题称为**初值问题**；另一种是给出积分曲线在

首末两端的性态,这类条件称为边界条件,对应的定解问题称为**边值问题**. 关于一阶常微分方程初值问题有下面的定理.

定理 10.2 对于常微分方程初值问题

$$\frac{\mathrm{d}y}{\mathrm{d}x}=f(x,y),y(x_0)=y_0, \tag{10.5}$$

如果 $x_0\in[a,b]$,函数 $f(x,y)$ 对 x 连续,且对 y 满足利普希茨(Lipschitz)条件,即存在常数 $L>0$,使

$$|f(x,y_1)-f(x,y_2)|\leqslant L|y_1-y_2| \tag{10.6}$$

对所有 $x\in[a,b]$ 及任意实数 y_1,y_2 都成立,则初值问题(10.1)和(10.3)在闭区间 $[a,b]$ 上有唯一解.

如果微分方程(组)难以求解或根本无解析解,则用微分方程(组)的数值解法. 所为数值解法就是求初值问题(10.5)的解 $y=y(x)$ 在一系列离散点

$$x_0<x_1<x_2<\cdots<x_n<\cdots$$

处 $y(x_n)$ 的近似值 $y_n(n=1,2,\cdots)$. 通常取等步长 h,则 $x_n=x_0+nh(n=1,2,\cdots)$.

建立数值解法,首先将微分方程离散化,一般经常采用数值微分法、数值积分法和泰勒(Taylor)逼近法等,下面简单介绍这些方法的基本思想.

10.2.1 数值微分法的基本思想

如果 $f(x,y)$ 中的 x 取小区间 $[x_n,x_{n+1}]$ 的左端点 x_n,用向前差商 $\frac{y(x_{n+1})-y(x_n)}{h}$ 代替 $y'(x_n)$ 代入式(10.5)中的导数,则(10.5)式中的微分方程化为

$$\frac{y(x_{n+1})-y(x_n)}{h}\approx f[x_n,y(x_n)]\ (n=0,1,2,\cdots),$$

化简得

$$y(x_{n+1})\approx y(x_n)+hf[x_n,y(x_n)]\ (n=0,1,2,\cdots).$$

如果用 $y(x_n)$ 的近似值 y_n 代入上式的右端,所得的结果 y_{n+1} 作为 $y(x_{n+1})$ 的近似值,则有

$$y_{n+1}=y_n+hf(x_n,y_n)\ (n=0,1,2,\cdots). \tag{10.7}$$

这样,求常微分方程初值问题(10.5)的数值解,可以通过求解

$$\begin{cases}y_{n+1}=y_n+hf(x_n,y_n)\ (n=0,1,2,\cdots),\\ y_0=y(x_0)\end{cases} \tag{10.8}$$

得到. 将初值 (x_0,y_0) 代入(10.7)式,得到 $y(x_1)$ 的近似值

$$y_1=y_0+hf(x_0,y_0).$$

再将(x_1, y_1)代入(10.7)式,得到$y(x_2)$的近似值

$$y_2 = y_1 + hf(x_1, y_1).$$

继续下去,可以逐次求出$y_1, y_2, \cdots$. (10.7)式称为**向前欧拉公式**. (10.8)式是个离散化的问题,也称为差分方程初值问题. 而(10.5)式中方程右端,已知函数$f(x,y)$中的x在小区间$[x_n, x_{n+1}]$上取不同的点,或用不同的差商近似导数公式,则将得到不同的公式.

10.2.2 数值积分法的基本思想

将初值问题(10.5)的解$y(x)$表示成积分形式,然后用数值积分方法离散化. 例如,将(10.5)式中的方程改写成$dy = f(x,y)dx$,等式两边在小区间$[x_n, x_{n+1}]$上积分得

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(x, y(x))\,dx \quad (n = 0,1,2,\cdots). \tag{10.9}$$

如果用y_n和y_{n+1}作为$y(x_n)$和$y(x_{n+1})$的近似值分别代入(10.9)式的左端,而右端的积分用左端点矩形公式

$$\int_{x_n}^{x_{n+1}} f[x, y(x)]\,dx \approx hf(x_n, y_n) \quad (n = 0,1,2,\cdots)$$

近似代替,则

$$y_{n+1} - y_n = hf(x_n, y_n) \quad (n = 0,1,2,\cdots),$$

移项后也得到向前欧拉公式(10.7),再代入初值问题(10.5)中,得(10.8)式. 用不同的数值积分公式,将得到不同的计算微分方程初值问题的数值公式.

10.2.3 泰勒(Taylor)多项式逼近法基本思想

如果将函数$y(x)$在小区间$[x_n, x_{n+1}]$的左端点x_n作泰勒展开,取一阶泰勒多项式逼近,则得

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + hy'(x_n) + \frac{1}{2}h^2 y''(\xi_n) \\ &\approx y(x_n) + hf(x_n, y(x_n)) \quad (n = 0,1,2,\cdots). \end{aligned}$$

如果用y_n和y_{n+1}作为$y(x_n)$和$y(x_{n+1})$的近似值分别代入上式,也得到向前欧拉公式(10.7),代入初值问题(10.5)中,得(10.8)式. $y(x)$在不同的点展开泰勒公式,将得到不同的公式.

以上三种方法都是将微分方程离散化的常用方法,取不同的点或多项式的次数,又可以得到不同类型的微分方程的数值计算公式. 其中泰勒多项式逼近法不仅可以推出数值计算公式,而且可以得到截断误差公式.

从另一种角度上说,常微分方程初值问题的数值解法分为两大类:单步法和

多步法. **单步法**是在计算 y_{n+1}时,只用到了前一步的值 y_n,x_n和 x_{n+1}. 因此,有了初值就可以往下计算,例如,龙格-库塔方法. **多步法**是在计算 y_{n+1}时,除了用到前一步的值 x_{n+1},x_n和 y_n以外,还要用到前面 k 步的值 $x_{n-j},y_{n-j}(j=1,2,\cdots,k)$. 例如,亚当斯方法.



习 题 10.2

1. 初值问题$\frac{dy}{dx}=ax+b,y(0)=0$有解 $y=0.5ax^2+bx$. 如果取 $x_n'=nh(n=0,1,2,\cdots)$,分别用数值微分法、数值积分法和泰勒多项式逼近法,推导出求解该初值问题的向前欧拉公式 $y_{n+1}=y_n+h(ax_n+b),n=0,1,2,\cdots$.

2. 用公式 $y_{n+1}=y_n+h(ax_n+b),n=0,1,2,\cdots$求解初值问题$\frac{dy}{dx}=ax+b,y(0)=0,a=b=1$在$[0,1]$上的数值解,取 $h=0.1$.

10.3 欧拉(Euler)方法及其 MATLAB 程序

欧拉方法是一种最简单的解微分方程的数值方法,其基本想法与数值微分法基本思想相同,是在小区间$[x_n,x_{n+1}]$上用差商$\frac{y(x_{n+1})-y(x_n)}{h}$代替(10.5)中方程左端的导数 y',而方程右端的已知函数$f(x,y)$中的 x 在小区间$[x_n,x_{n+1}]$上取值不同,则有以下不同的方法. 本节主要介绍向前(向后)欧拉公式和几何意义及其误差估计,用向前欧拉公式计算初值问题的三种 MATLAB 程序和向后欧拉公式的 MATLAB 程序及其具体的算例.

10.3.1 向前欧拉公式及其误差估计

(一) 欧拉方法

如果$f(x,y)$中的 x 取小区间$[x_n,x_{n+1}]$的左端点 x_n,由上节用三种不同的方法得到相同的**向前欧拉公式**

$$\begin{cases}y_{n+1}=y_n+hf(x_n,y_n),n=0,1,\cdots,\\x_n=x_0+nh.\end{cases}\tag{10.10}$$

欧拉方法就是用差分方程初值问题

$$\begin{cases}y_{n+1}=y_n+hf(x_n,y_n)(n=0,1,2,\cdots),\\y_0=y(x_0)\end{cases}$$

的解作为常微分方程初值问题(10.5)的数值解,即将初值 y_0代入(10.7)式,逐

次求出 $y_1,y_2,\cdots$.

向前欧拉公式的几何意义如图10－1所示.方程(10.5)的精确解 $y=y(x)$ 称积分曲线,曲线上任一点 (x,y) 的切线斜率等于已知函数 $f(x,y)$. 曲线 $y=y(x)$ 和数值解都是从初值点 $P_0(x_0,y_0)$ 开始,用该点的斜率 $f(x_0,y_0)$ 作一直线段,在 $x=x_1$ 处得到 $P_1(x_1,y_1)$ 点,其中 $y_1=y_0+hf(x_0,y_0)$. 再从 P_1 出发,以 $f(x_1,y_1)$ 为斜率作直线段推进到 $x=x_2$ 处得到 $P_2(x_2,y_2)$ 点,其中 $y_2=y_1+hf(x_1,y_1)$. 继续下去,得到的折线 $P_0P_1P_2\cdots$ 可作为积分曲线 $y=y(x)$ 的近似.

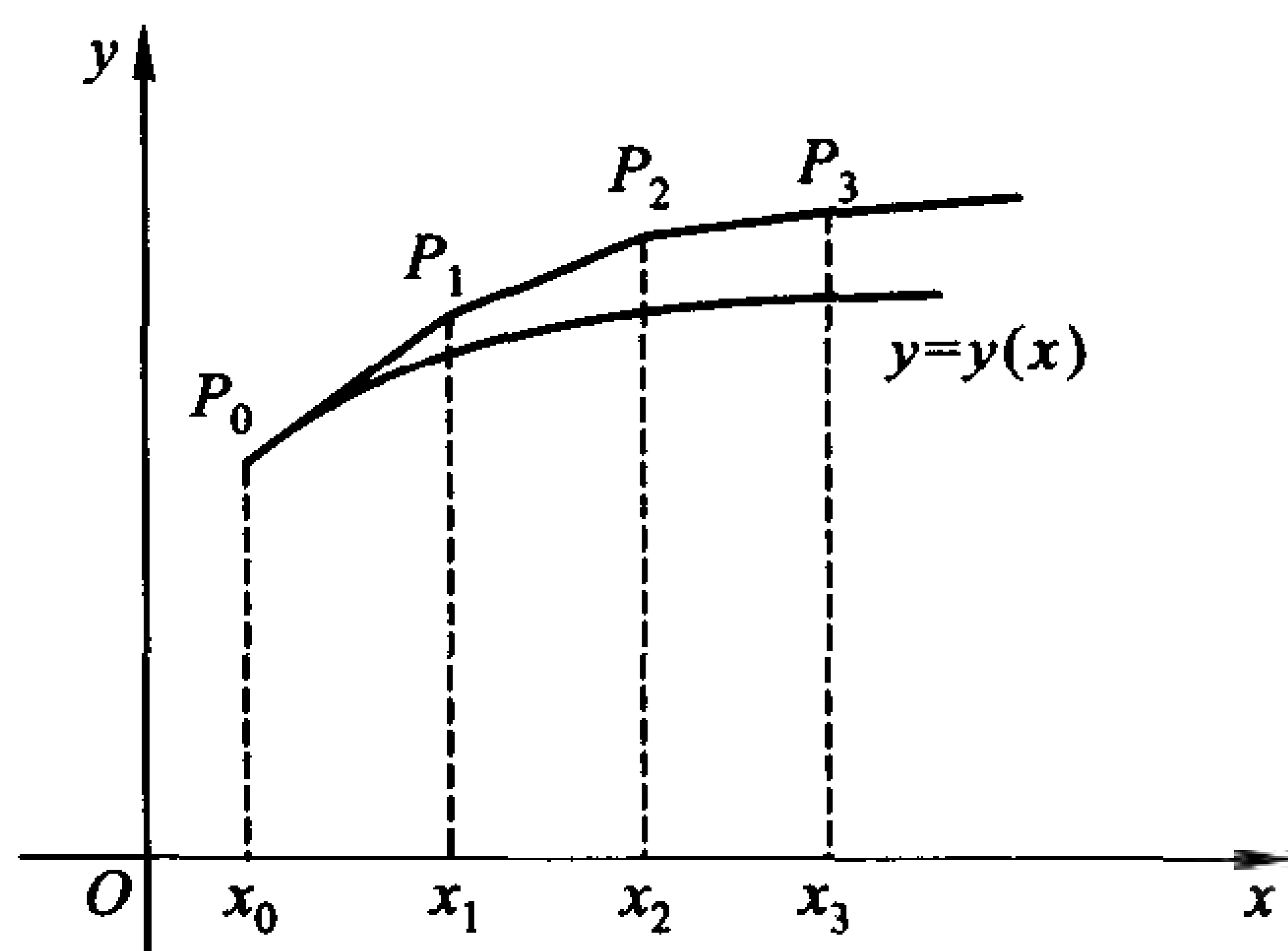


图 10－1　向前欧拉公式的几何意义

例 10.3.1　用欧拉方法求初值问题

$$\begin{cases}\dfrac{dy}{dx}=x-y, & 0\leqslant x\leqslant 1,\\ y|_{x=0}=1,\end{cases}$$

的数值解,分别取 $h=0.075\ 0, 0.007\ 5$,并计算误差,画出精确解和数值解的图形.

解　(1) 先求精确解.

输入程序

```
>> y=dsolve('(Dy)+y-x=0','y(0)=1','x')
```

运行后屏幕显示常微分方程在给定初始条件下的精确解 y 如下

```
y =x-1+2*exp(-x)
```

(2) 用欧拉方法求初值问题的数值解,并计算误差,画出精确解和数值解的图形.

① 对此问题的向前欧拉公式(10.10)的具体形式为

$$\begin{cases}y_{n+1}=y_n+h(x_n-y_n), n=0,1,\cdots,\\ x_n=nh,\end{cases}$$

② 编写并保存名为 Eulerli1.m 的 MATLAB 计算和画图的主程序如下

```
function P=Eulerli1(x0,y0,b,h)
n=(b-x0)/h; X=zeros(n,1);
Y=zeros(n,1); k=1; X(k)=x0; Y(k)=y0;
for k=1:n
  X(k+1)=X(k)+h; Y(k+1)=Y(k)+h*(X(k)-Y(k)); k=k+1;
end
y=X-1+2*exp(-X); plot(X,Y,'mp',X,y,'b-')
```

```
grid,xlabel('自变量 X'), ylabel('因变量 Y')
title('用向前欧拉公式求 dy/dx=x-y,y(0)=1 在[0,1]上的数值解和精确解 y=x-1+2 exp(-x)')
legend('h=0.075 时,dy/dx=x-y,y(0)=1 在[0,1]上的数值解','精确解 y=x-1+2 exp(-x)')
jwY=y-Y;xwY=jwY./y;
k1=1:n;k=[0,k1]; P=[k',X,Y,y,jwY,xwY];
```

③ 在 MATLAB 工作窗口输入下面的程序

```
>>x0=0;y0=1;b=1;h=0.0750;
P=Eulerli1(x0,y0,b,h)
```

运行后屏幕显示取 $h=0.0750$ 时,该初值问题的精确解 y、用欧拉方法求此初值问题与自变量 X 对应的数值解 Y,Y 的绝对误差 jwY 和相对误差 xwY(见表 10-1),精确解 y 和数值解 Y 的图形(见图 10-2).

表 10-1 取 $h=0.0750$ 时与 X 对应的精确解 y、数值解 Y、绝对误差 jwY 和相对误差 xwY

k	X	数值解 Y	精确解 y	绝对误差 jwY	相对误差 xwY
0	0	1.000 0	1.000 0	0	0
1.000 0	0.075 0	0.925 0	0.930 5	0.005 5	0.005 9
2.000 0	0.150 0	0.861 3	0.871 4	0.010 2	0.011 7
3.000 0	0.225 0	0.807 9	0.822 0	0.014 1	0.017 2
4.000 0	0.300 0	0.764 2	0.781 6	0.017 4	0.022 3
5.000 0	0.375 0	0.729 4	0.749 6	0.020 2	0.027 0
6.000 0	0.450 0	0.702 8	0.725 3	0.022 5	0.031 0
7.000 0	0.525 0	0.683 8	0.708 1	0.024 3	0.034 3
8.000 0	0.600 0	0.671 9	0.697 6	0.025 7	0.036 8
9.000 0	0.675 0	0.666 5	0.693 3	0.026 8	0.038 6
10.000 0	0.750 0	0.667 2	0.694 7	0.027 6	0.039 7
11.000 0	0.825 0	0.673 4	0.701 5	0.028 1	0.040 0
12.000 0	0.900 0	0.684 7	0.713 1	0.028 4	0.039 8
13.000 0	0.975 0	0.700 9	0.729 4	0.028 5	0.039 1

④ 在 MATLAB 工作窗口输入下面的程序

```
>>h1=0.0075; P1=Eulerli1(x0,y0,b,h1)
legend('h1=0.0075 时,dy/dx=x-y,y(0)=1 在[0,1]上的数值解','精确解 y=x-1+2 exp(-x)')
```

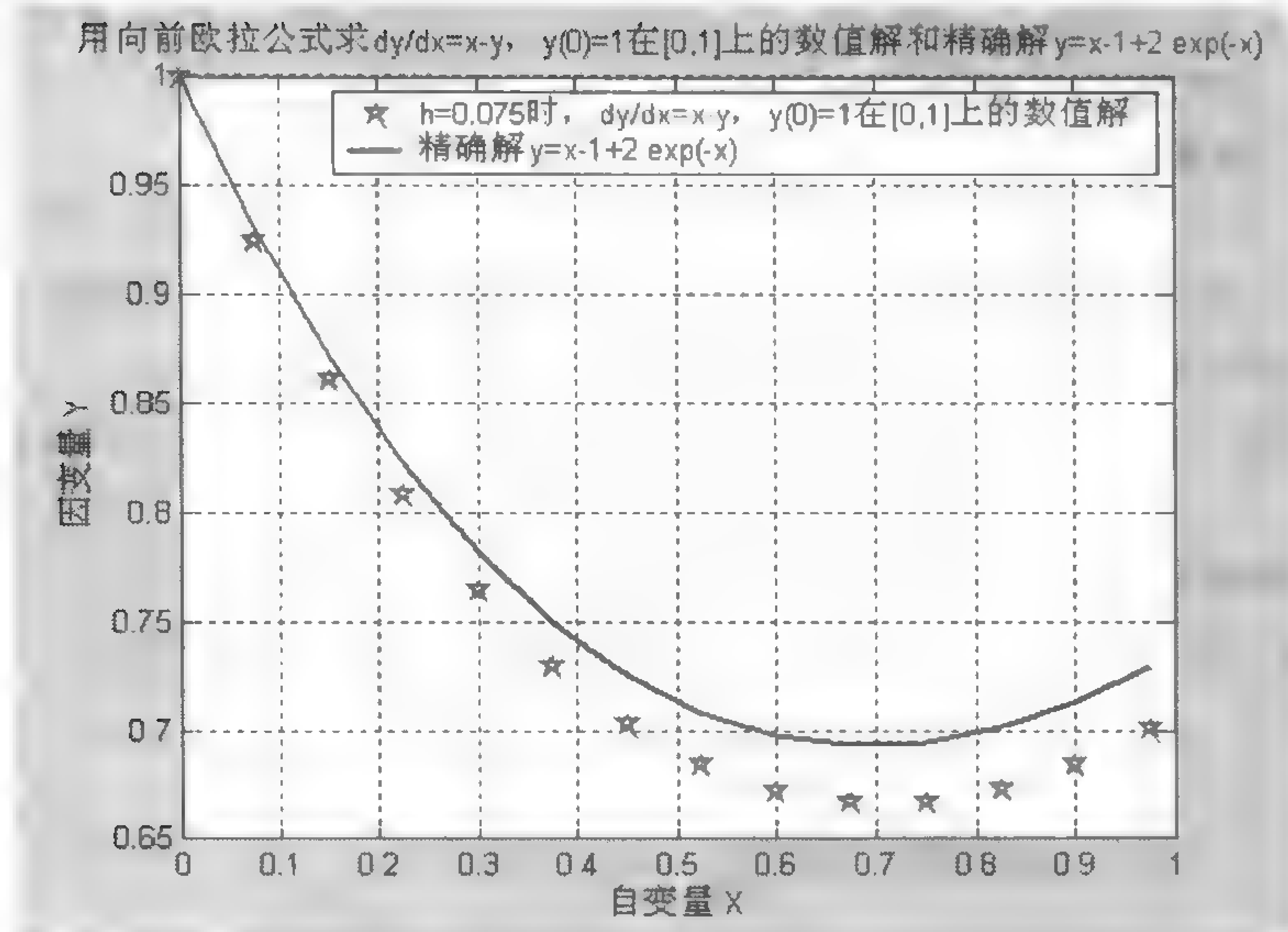


图 10-2 取 $h=0.0750$ 时的精确解和数值解

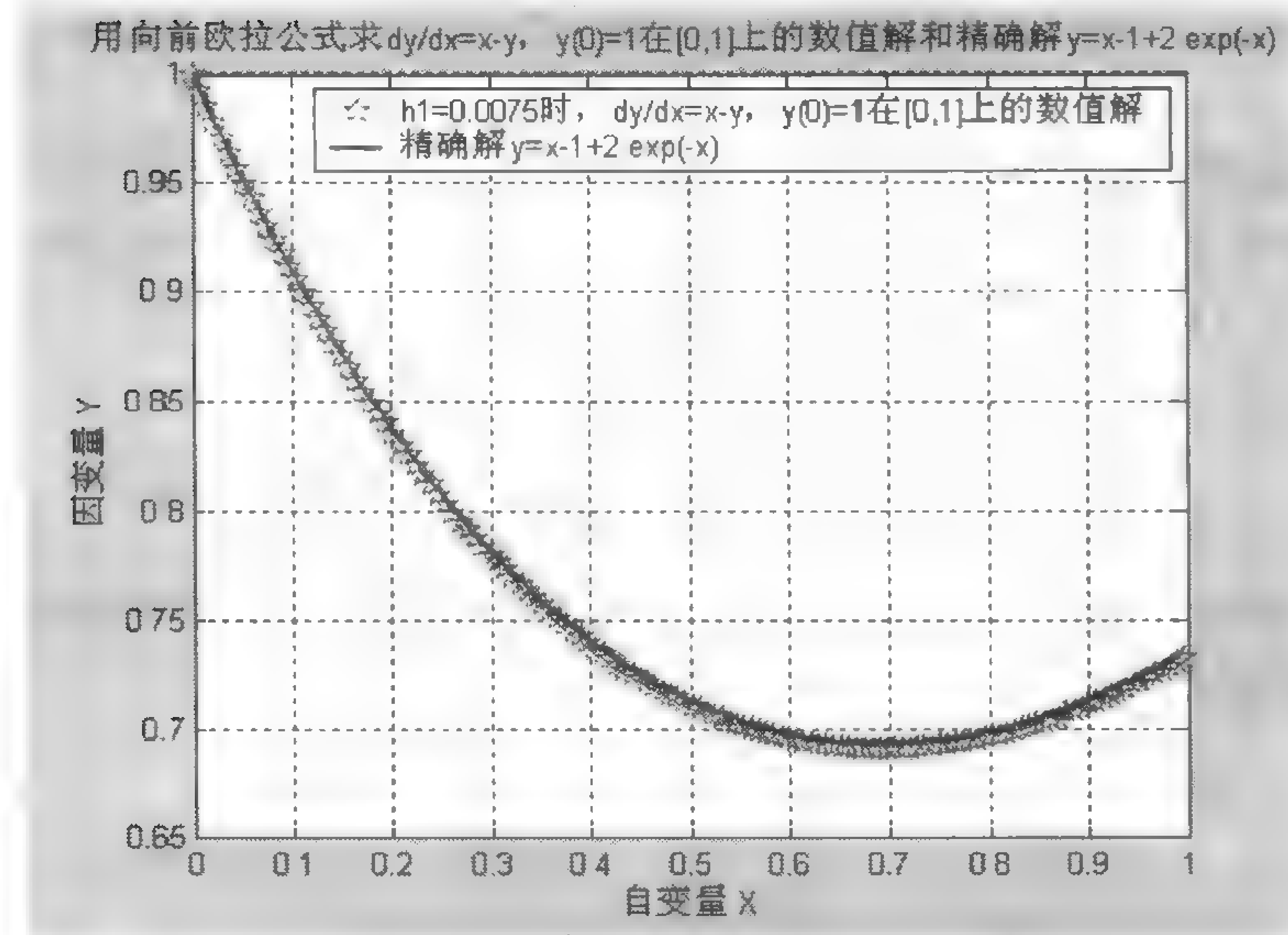


图 10-3 取 $h=0.0075$ 时的精确解和数值解

```
    title('用向前欧拉公式求 dy/dx - x - y,y(0) =1 在[0,1]上的数值解和精
确解 y = x - 1 + 2exp( - x)')
```

运行后屏幕显示取 $h=0.0075$ 时，该初值问题的精确解 y、用欧拉方法求此初值问题与自变量 X 对应的数值解 Y，Y 的绝对误差 jwY 和相对误差 xwY（略），精确

解 y 和数值解 Y 的图形(见图 10-3). 由图 10-2 和图 10-3 可以看出, h 的绝对值越小,数值解 Y 与精确解 y 的误差越小.

(二) 误差估计

当 $n=1,2,\cdots$ 时,向前欧拉公式(10.8)右端的 y_n 都是近似的,只有当 $n=0$ 时,(10.7)式右端的 y_0 是精确的,所以用(10.8)式计算的 y_{n+1} 会有累积误差,分析累积误差比较复杂,这里先讨论比较简单的所谓局部截断误差.

假定用(10.7)式时,其右端的 y_n 没有误差,即 $y_n=y(x_n)$,那么由此算出

$$y_{n+1}=y(x_n)+hf(x_n,y(x_n)). \tag{10.11}$$

局部截断误差指的是,按(10.11)式计算由 x_n 到 x_{n+1} 这一步的计算值 y_{n+1} 与精确值 $y(x_{n+1})$ 之差 $y(x_{n+1})-y_{n+1}$. 为了估计它,由泰勒展开得到的精确值 $y(x_{n+1})$ 是

$$y(x_{n+1})=y(x_n)+hy'(x_n)+\frac{h^2}{2}y''(x_n)+O(h^3). \tag{10.12}$$

(10.11)、(10.12)二式相减(注意到 $y'=f(x,y)$)得

$$y(x_{n+1})-y_{n+1}=\frac{h^2}{2}y''(x_n)+O(h^3)\cong O(h^2), \tag{10.13}$$

即局部截断误差是 h^2 阶的,而数值算法的精度定义如下:

定义 10.1 若一种算法的局部截断误差为 $O(h^{p+1})$,则称该算法具有 p 阶**精度**.

向前欧拉公式的精度为 1 阶. 由泰勒多项式逼近法可得到下面的定理.

定理 10.3 设函数 $y(x)$ 是常微分方程初值问题(10.5)的解,如果 $y(x)$ 在区间 $[a,b]$ 上具有二阶连续导数,且由向前欧拉公式

$$\begin{cases} y_{n+1}=y_n+hf(x_n,y_n), n=0,1,\cdots, \\ x_n=x_0+nh \end{cases}$$

计算(10.5)式的数值解的序列为 $\{(x_n,y_n)\}|_{n=0}^{m}$,则

(1) 向前欧拉公式在子区间 $[x_n,x_{n+1}]\subseteq[a,b]$ 上的局部截断误差是 h^2 阶的,且截断误差公式为

$$E(f,y_n)=y(x_{n+1})-y_{n+1}=\frac{1}{2}h^2y''(\xi_n)\ (n=0,1,2,\cdots), \tag{10.14}$$

其中 $\xi_n\in[x_n,x_{n+1}]$.

(2) 向前欧拉公式在 $[a,b]$ 上的截断误差是 h 阶的.

10.3.2 向前欧拉方法的三种 MATLAB 程序

下面分别介绍三种用向前欧拉公式求解常微分方程初值问题(10.5)的数值解及其截断误差公式的 MATLAB 程序,并且通过例题说明这三种程序的应用.

(一) 向前欧拉公式的 MATLAB 主程序 1

下面介绍根据向前欧拉公式(10.8)编写的数值计算常微分方程初值问题(10.5)的 MATLAB 程序.

> **用向前欧拉公式(10.8)求解常微分方程初值问题(10.5)的数值解及其截断误差公式的 MATLAB 主程序 1**
>
> 输入量:*funfcn* 是函数 $f(x,y)$,x_0 和 y_0 是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,n 是自变量 x 取值的个数,*tol* 是精度.
>
> 输出量:数组 X 的元素是由自变量 x 的取值组成,数组 Y 的元素是利用向前欧拉公式(10.8)求出常微分方程初值问题(10.5)在 X 的元素处的数值解,*REn* 是在每个子区间 $[x_k,x_{k+1}]$ 上的局部截断误差公式,h 是步长,其中 *dy2* 是 $y(x)$ 的 2 阶导数 $y''(x)$.

```
function [h,k,X,Y,P,REn] = Qeuler1(funfcn,x0,y0,b,n,tol)
x = x0; h = (b - x)/n; X = zeros(n,1); y = y0;
Y - zeros(n,1); k = 1; X(k) = x; Y(k) = y;
for k = 2:n + 1
      fxy - feval(funfcn,x,y); delta = norm(h * fxy,'inf');
      wucha = tol * max(norm(y,'inf'),1.0);
  if delta > = wucha
     x = x + h; y = y + h * fxy; X(k) = x;Y(k) = y';
  end
    plot(X,Y,'rp')
    grid,xlabel('自变量 X'), ylabel('因变量 Y')
    title('用向前欧拉公式计算 dy/dx = f(x,y),y(x0) = y0 在[x0,b]上
的数值解')
end
P = [X,Y]; syms dy2,REn = 0.5 * dy2 * h^2;
```

例 10.3.2 用向前欧拉公式(10.8)求解初值问题

$$\frac{dy}{dx}=-3y+8x-7,y(0)=1,$$

分别取 $n=10,100$,并将计算结果与精确解作比较,写出在每个子区间 $[x_k,x_{k+1}]$ 上的局部截断误差公式,画出数值解与精确解在区间 $[0,1]$ 上的图形.

解 (1) 建立并保存名为 funfcn.m 的 M 文件函数.

```
function f - funfcn (x,y)
         f = 8 * x - 3 * y - 7;
```

(2) 建立并保存名为 Qeuler1.m 的 M 文件函数.

(3) 输入程序

```
>> S1 = dsolve('Dy = 8*x - 3*y - 7','y(0) = 1','x')
```

运行后屏幕显示结果如下

```
S1 =
   8/3*x - 29/9 + 38/9*exp(-3*x)
```

(4) 输入程序

```
>> subplot(2,1,1)
x0 = 0;y0 = 1;b = 1 - 1.e - 4;n = 100;tol = 1.e - 4;
[h1,k1,x1,Y1,P1,Ren1] = QEuler1(@funfcn,x0,y0,b,n,tol)
hold on
S1 = 8/3*x1 - 29/9 + 38/9*exp(-3*x1), plot(x1,S1,'b-')
title('用向前欧拉公式计算 dy/dx = 8x - 3y - 7,y(0) = 1 在[0,1]上的数值解')
legend('n = 100 时,dy/dx = 8x - 3y - 7,y(0) = 1 在[0,1]上的数值解','
dy/dx = 8x - 3y - 7,y(0) = 1 在[0,1]上的精确解')
hold off
jdwucl = S1 - Y1; jwY1 = S1 - Y1;xwY1 = jwY1./S1;k1 = 1:n;k = [0,k1];
P1 = [k',x1,Y1,S1,jwY1,xwY1]
subplot(2,1,2)
n1 = 10;
[n2,k2,x2,Y2,P2,Ren2] = QEuler1(@funfcn,x0,y0,b,n1,tol)
hold on
S1 = 8/3*x2 - 29/9 + 38/9*exp(-3*x2), plot(x2,S1,'b-')
legend('n = 10 时,dy/dx = 8x - 3y - 7,y(0) = 1 在[0,1]上的数值解','
dy/dx = 8x - 3y - 7,y(0) = 1 在[0,1]上的精确解')
hold off
jwY2 = S1 - Y2;xwY2 = jwY2./S1;k1 = 1:n1;
k = [0,k1]; P2 = [k',x2,Y2,S1,jwY2,xwY2]
```

运行后屏幕显示分别取 $n = 10,100$ 时,所给的初值问题在[0,1]上的自变量 x 的值构成的数组 $X_i(i = 1,2)$,利用向前欧拉公式(10.8)求出的与 $X_i(i = 1,2)$ 对应的数值解 $Y_i(i = 1,2)$,Y_i 的绝对误差 $jwY_i(i = 1,2)$ 和相对误差 $xwY_i(i = 1,2)$(见表 10-2),每个子区间$[x_k,x_{k+1}]$上的局部截断误差公式 $Ren_2 = 1/200 * dy2$,数值解 Y 与精确解的图形(见图 10-4).

表 10-2 取 $n = 10$ 时,初值问题在 X_1 处的精确解 y、数值解 Y_1、绝对误差 jwY_1 和相对误差 xwY_1

k	自变量 X_1	数值解 Y_1	精确解 y	绝对误差 jwY_1	相对误差 xwY_1
0	0	1.000 0	1.000 0	0	0
1	0.100 0	0.000 1	0.172 4	0.172 3	0.999 4
2	0.200 0	-0.619 9	-0.371 6	0.248 3	-0.668 1
3	0.300 0	-0.973 9	-0.705 5	0.268 4	-0.380 4

续表

k	自变量 X_i	数值解 Y_i	精确解 y	绝对误差 jwY_i	相对误差 xwY_i
4	0.400 0	-1.141 7	-0.883 8	0.257 9	-0.291 8
5	0.500 0	-1.179 2	-0.946 8	0.232 5	-0.245 5
6	0.599 9	-1.125 5	-0.924 3	0.201 2	-0.217 7
7	0.699 9	-1.007 9	-0.838 6	0.169 3	-0.201 9
8	0.799 9	-0.845 6	-0.706 0	0.139 6	-0.197 8
9	0.899 9	-0.652 0	-0.538 6	0.113 4	-0.210 5
10	0.999 9	-0.436 5	-0.345 5	0.091 0	-0.263 2

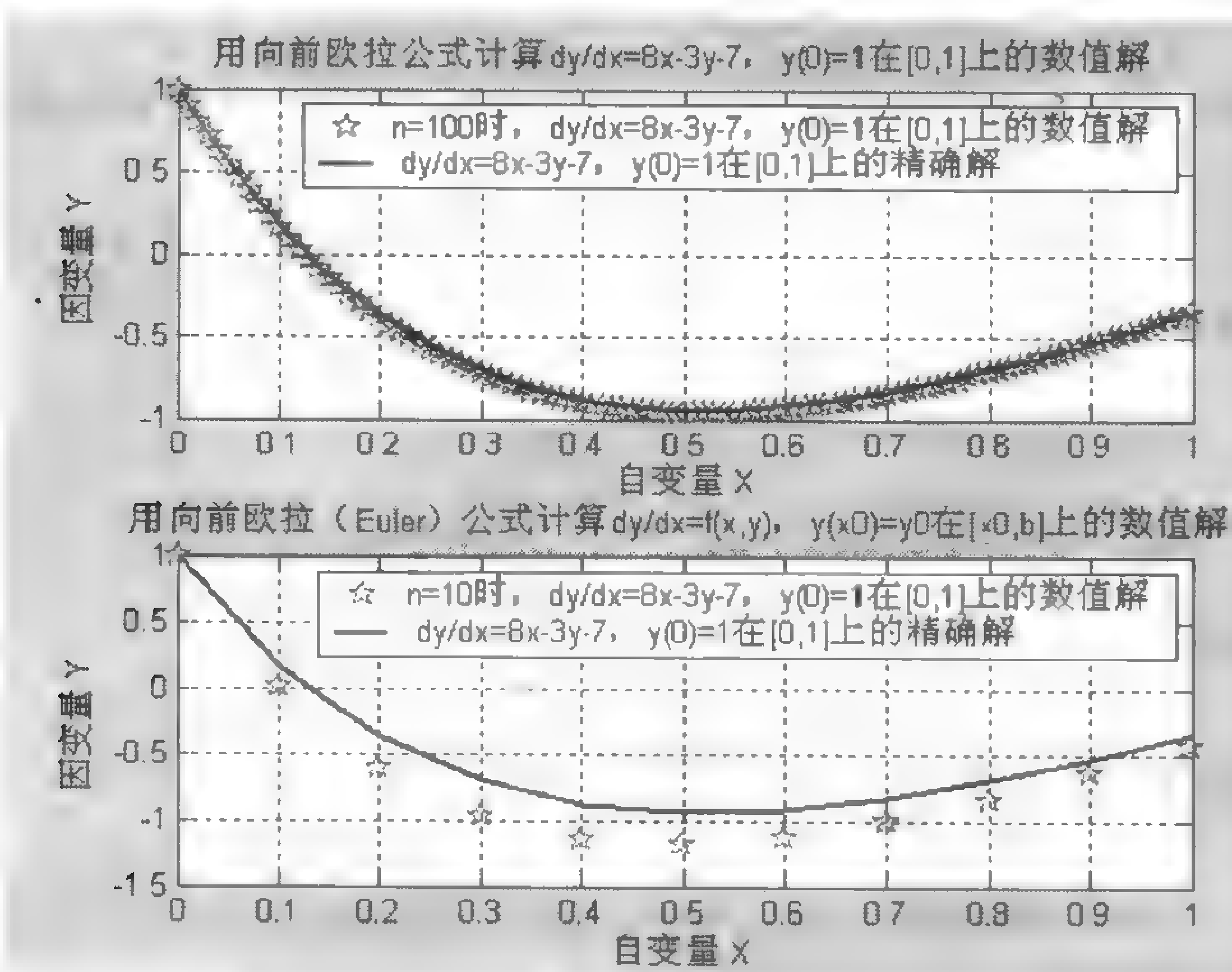


图 10-4　$n=10,100$ 时，$\frac{dy}{dx}=-3y+8x-7, y(0)=1$ 在[0,1]上的数值解和精确解

由图 10-4 和表 10-2 可以看出，取 $n=10$ 时，利用向前欧拉公式(10.8)求出的该初值问题在[0,1]上的数值解 Y_i 与精确解 y 的绝对误差向量 jwY_i 和相对误差向量 xwY_i 较大，但是当 $n=100$ 时，数值解 Y_i 的图形和精确解 y 的图形几乎重合，即节点的个数 n 越多，误差越小. 这说明当节点的个数 n 足够大时，这种方法是十分有效的.

例 10.3.3　分别用向前欧拉公式(10.8)、一、四阶泰勒逼近及用 MATLAB 函数 double 求解初值问题

$$\begin{cases}\frac{dy}{dx}=1-y\cos x, 0\leqslant x\leqslant 2,\\ y(0)=0.\end{cases}$$

分别取 $n=40,80$ 时,并将计算结果与精确解作比较,写出在每个子区间$[x_k, x_{k+1}]$上的局部截断误差公式,画出数值解与精确解在区间[0,2]上的图形.

解 (1) 建立并保存名为 funfcn.m 的 M 文件函数

```
function f = funfcn (X,y)
        f =1 -y * cos(X);
```

(2) 输入程序

```
>> S1 =dsolve('Dy =1 -y * cos(X)','y(0) =0 ','X')
```

运行后屏幕显示结果

```
S1 =
     exp( -sin(X)) * Int(exp(sin(z1)),z1 = 0 .. X)
```

运行后屏幕显示结果说明精确解 $y=e^{-\sin x}\int_0^x e^{\sin u}du$ 不能表示为初等函数.

因为 $\int_0^x e^{\sin u}dx=\int_0^x[1+\sin u+o(\sin u)]du=1+x-\cos x+\int_0^x o(\sin u)du.$

所以,精确解

$$\begin{aligned} y &= e^{-\sin x}\int_0^x e^{\sin u}du \\ &= e^{-\sin x}(1+x-\cos x)+e^{-\sin x}\int_0^x o(\sin u)du\to 0\ (x\to 0^+). \end{aligned}$$

取一阶泰勒逼近

$$y\approx e^{-\sin x}(1+x-\cos x).$$

同理,取四阶泰勒逼近,输入程序

```
>> syms X
y = exp( -sin(X)) * int(1 +sin(u) +((sin(u))^2)/2 +((sin(u))^
3)/6 +((sin(u))^4)/24 ,u, 0 ,X)
```

根据运行后,将屏幕显示结果整理得四阶泰勒逼近为

$$y\approx e^{\sin x}\left[\frac{10}{9}+\frac{81}{64}x-\cos x\left(\frac{10}{9}+\frac{17}{64}\sin x+\frac{1}{18}\sin^2 x+\frac{1}{96}\sin^3 x\right)\right].$$

(3) 取 $n=40$ 时,输入程序

```
>> syms u
h =2 /40,X =0:h:2;
S =exp( -sin(X)) * int(exp(sin(u)),u,0,X); S1 = double(S);
subplot(4,1,1)
plot(X,S1,'bo'), grid
legend('用 double 计算 dy/dx =1 -y cos(X),y(0) =0 在[0,2]上的精
```

```
确解的数值解')
        subplot(4,1,2)
        y=exp(-sin(X)).*(1+X-cos(X)); plot(X,y,'g-'), grid
        legend('y=exp(-sin(x))(1+x-cos(x))在[0,2]上图形')
        subplot(4,1,3)
        y1 = exp( - sin(X)).* ( 81/64 * X -10/9 * cos(X) -17/64 *
cos(X).*sin(X) -1/18 *sin(X).^2.*cos(X) -1/96 *sin(X).^3.*cos(X) +
10/9); plot(X,y1,'m-'), grid
        legend('四阶泰勒逼近的解在[0,2]上图形')
        subplot(4,1,4)
        x0=0;y0=0;b=2-1.e-4;n=40;tol=1.e-4;
        [h,k,X,Y,P,Ren]=QEuler1(@funfcn,x0,y0,b,n,tol)
        legend('n=40时,用向前欧拉公式计算 dy/dx=1-y cos(X),y(0)=0 在
[0,2]上的数值解')
        syms u
        X =0:0.05:2; S=exp(-sin(X))*int(exp(sin(u)),u,0,X);
        S1=double(S);jwS1y1= y1'-S1',
        xwS1y1=jwS1y1./S1', jwYy1= y1'-Y, xwYy1=jwYy1./Y,
        k1=1:n;k=[0,k1]; P=[k',X',Y,y1',jwYy1,xwYy1],
        PS=[k',X',S1',jwS1y1],xwS1y1
```

运行后屏幕显示取 $n=40$ 时,分别用向前欧拉公式(10.8)和一、四阶泰勒多项式逼近法及 double 求解此初值问题在[0,2]上的与自变量 X 对应的数值解 Y,y,y_1,S_1 及其图形(见图 10-5(a)),Y 与 y_1,S_1 与 y_1 的绝对误差向量 $jwYy_1$,xwS_1y_1 和相对误差向量 $xwYy_1$,xwS_1y_1(见表 10-3),向前欧拉公式在每个子区间$[x_k,x_{k+1}]$上的局部截断误差公式为

$$Ren\approx 0.001\,2y''(\xi_k),\xi_k\in[x_k,x_{k+1}],k=1,2,\cdots,40.$$

表 10-3 取 $n=40$ 时,用向前欧拉公式和四阶泰勒逼近法及 double 求初值问题的结果

k'	X'	Y	Y_1	S_1	$JwYy_1$	jwS_1y_1	$XwYy_1$	xwS_1y_1
0	0	0	0	0.691 2e-093	0	-0.000 0	NaN	-0.000 0e+093
1	0.050 0	0.050 0	0.048 8	0.657 5e-093	-0.001 2	0.048 8	-0.024 5	0.074 2e+093
2	0.100 0	0.097 5	0.095 2	0.625 5e-093	-0.002 3	0.095 2	-0.023 8	0.152 1e+093
3	0.150 0	0.142 6	0.139 3	0.595 3e-093	-0.003 3	0.139 3	-0.023 1	0.234 1e+093
4	0.200 0	0.185 6	0.181 4	0.566 7e-093	-0.004 1	0.181 4	-0.022 3	0.320 2e+093
5	0.250 0	0.226 5	0.221 6	0.539 7e-093	-0.004 9	0.221 6	-0.021 5	0.410 6e+093
6	0.300 0	0.265 5	0.260 0	0.514 4e-093	-0.005 5	0.260 0	-0.020 7	0.505 5e+093
7	0.350 0	0.302 8	0.296 8	0.490 6e-093	-0.006 0	0.296 8	-0.019 8	0.605 1e+093

续表

k'	X'	Y	Y_1	S_1	$JwY\gamma_1$	$jwS_1\gamma_1$	$XwY\gamma_1$	$xwS_1\gamma_1$
8	0.400 0	0.338 6	0.332 2	0.468 3e-093	-0.006 4	0.332 2	-0.018 9	0.709 4e+093
9	0.450 0	0.373 0	0.366 3	0.447 4e-093	-0.006 7	0.366 3	-0.018 0	0.818 7e+093
10	0.500 0	0.406 2	0.399 3	0.428 0e-093	-0.007 0	0.399 3	-0.017 1	0.933 0e+093
11	0.550 0	0.438 4	0.431 3	0.409 8e-093	-0.007 1	0.431 3	-0.016 2	1.052 3e+093
12	0.600 0	0.469 7	0.462 5	0.393 0e-093	-0.007 2	0.462 5	-0.015 3	1.176 9e+093
13	0.650 0	0.500 3	0.493 1	0.377 4e-093	-0.007 2	0.493 1	-0.014 4	1.306 7e+093
14	0.700 0	0.530 4	0.523 3	0.362 9e-093	-0.007 1	0.523 3	-0.013 4	1.441 8e+093
15	0.750 0	0.560 1	0.553 1	0.349 6e-093	-0.007 0	0.553 1	-0.012 5	1.582 1e+093
16	0.800 0	0.589 6	0.582 8	0.337 3e-093	-0.006 8	0.582 8	-0.011 6	1.727 6e+093
17	0.850 0	0.619 1	0.612 5	0.326 1e-093	-0.006 6	0.612 5	-0.010 7	1.878 3e+093
18	0.900 0	0.648 6	0.642 3	0.315 8e-093	-0.006 3	0.642 3	-0.009 8	2.033 9e+093
19	0.950 0	0.678 5	0.672 5	0.306 4e-093	-0.006 0	0.672 5	-0.008 9	2.194 5e+093
20	1.000 0	0.708 7	0.703 1	0.298 0e-093	-0.005 7	0.703 1	-0.008 0	2.359 7e+093
21	1.050 0	0.739 6	0.734 4	0.290 3e-093	-0.005 2	0.734 4	-0.007 1	2.529 4e+093
22	1.100 0	0.771 2	0.766 4	0.283 5e-093	-0.004 8	0.766 4	-0.006 2	2.703 3e+093
23	1.150 0	0.803 7	0.799 4	0.277 5e-093	-0.004 3	0.799 4	-0.005 3	2.881 2e+093
24	1.200 0	0.837 3	0.833 6	0.272 2e-093	-0.003 7	0.833 6	-0.004 4	3.062 8e+093
25	1.250 0	0.872 1	0.869 0	0.267 6e-093	-0.003 1	0.869 0	-0.003 6	3.247 5e+093
26	1.300 0	0.908 4	0.905 9	0.263 7e-093	-0.002 4	0.905 9	-0.002 7	3.435 2e+093
27	1.350 0	0.946 2	0.944 5	0.260 5e-093	-0.001 7	0.944 5	-0.001 8	3.625 4e+093
28	1.400 0	0.985 8	0.985 0	0.258 0e-093	-0.000 9	0.985 0	-0.000 9	3.817 7e+093
29	1.450 0	1.027 5	1.027 5	0.256 1e-093	0.000 1	1.027 5	0.000 1	4.011 5e+093
30	1.500 0	1.071 3	1.072 3	0.254 9e-093	0.001 1	1.072 3	0.001 0	4.206 5e+093
31	1.550 0	1.117 5	1.119 6	0.254 3e-093	0.002 2	1.119 6	0.001 9	4.402 2e+093
32	1.600 0	1.166 3	1.169 7	0.254 4e-093	0.003 4	1.169 7	0.002 9	4.598 1e+093
33	1.650 0	1.218 0	1.222 8	0.255 1e-093	0.004 8	1.222 8	0.003 9	4.793 7e+093
34	1.700 0	1.272 8	1.279 1	0.256 4e-093	0.006 3	1.279 1	0.005 0	4.988 6e+093
35	1.750 0	1.331 0	1.339 0	0.258 4e-093	0.008 0	1.339 0	0.006 0	5.182 2e+093
36	1.800 0	1.392 8	1.402 7	0.261 0e-093	0.009 9	1.402 7	0.007 1	5.374 1e+093
37	1.850 0	1.458 7	1.470 7	0.264 3e-093	0.012 0	1.470 7	0.008 2	5.563 9e+093
38	1.900 0	1.528 7	1.543 1	0.268 3e-093	0.014 3	1.543 1	0.009 4	5.751 1e+093
39	1.950 0	1.603 4	1.620 4	0.273 0e-093	0.016 9	1.620 4	0.010 6	5.935 4e+093
40	2.000 0	1.683 1	1.702 9	0.278 4e-093	0.019 8	1.702 9	0.011 8	6.116 3e+093

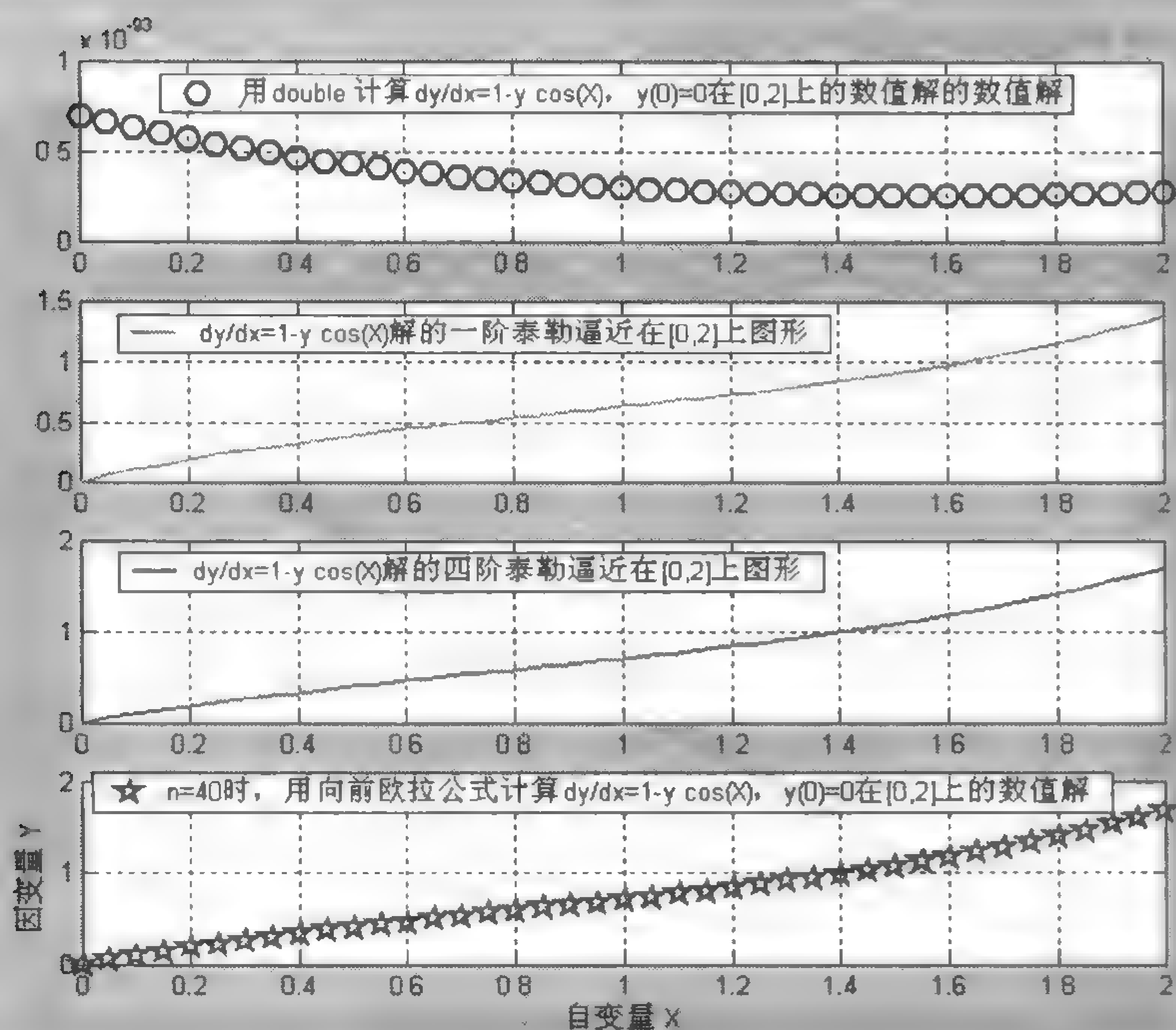


图 10-5(a) $n=40$,用向前欧拉公式和一、四阶泰勒逼近法及 double 求初值问题的图

取 $n=40$ 时,输入程序

```
>>syms u
X =0:0.05:2;
S=exp(-sin(X))*int(exp(sin(u)),u,0,X); S1 = double(S);
y=exp(-sin(X)).*(1+X-cos(X)); plot(X,y,'g-'), grid
x0 =0;y0 =0;b=2 -1.e-4;n=40;tol =1.e-4;
[h,k,X,Y,P,Ren] =QEuler1(@funfcn,x0,y0,b,n,tol)
y4 =exp(-sin(X)).*(81/64*X-10/9*cos(X) -17/64*cos(X).
*sin(X) -1/18*sin(X).^2.*cos(X) -1/96*sin(X).^3.*cos(X) +10/9);
plot(X,S1,'mo',X,y,'g-.',X,Y,'rp',X,y4,'b-'), grid
legend('用 double 计算 dy/dx=1-y cos(X),y(0)=0 在[0,2]上的精确
确的数值解','y=exp(-sin(X))(1+X-cos(X))在[0,2]上图形','n=40 时,用向前
欧拉公式计算 dy/dx=1-y cos(X),y(0)=0 在[0,2]上的数值解','y4=exp(-sin
(X))(10/9+81/64X-cos(X)(10/9+17/64 sin(X) +1/18 sin(X)^2 +1/96
sin(X)^3))在[0,2]上图形')
```

运行后屏幕显示取 $n=40$ 时,分别用向前欧拉公式(10.8)和一、四阶泰勒多项

式逼近法及 double 求此解初值问题在[0,2]上的与自变量 X 对应的数值解 Y, y, y_1, S_1 及其图形(见图 10 - 5(b))

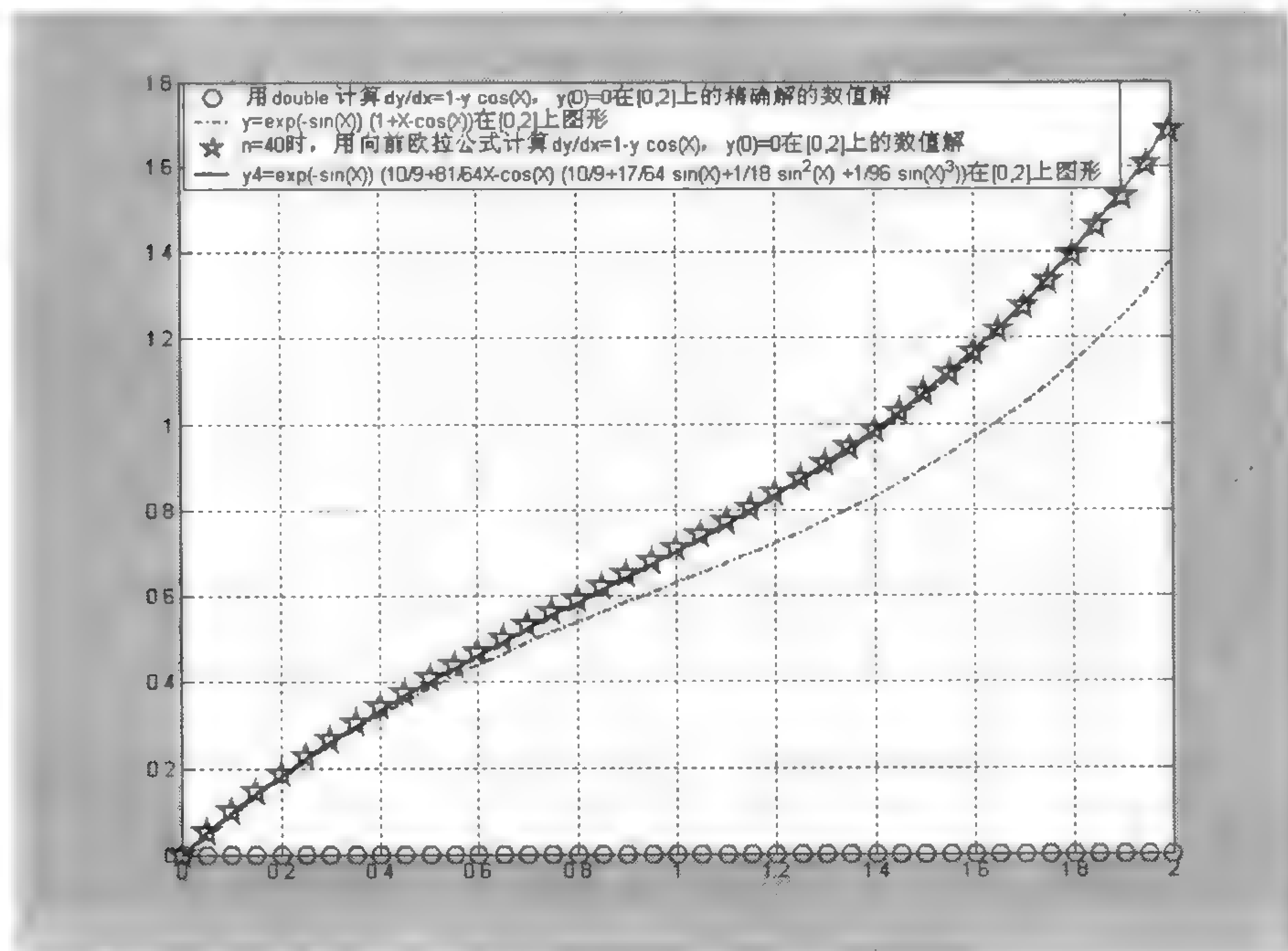


图 10 - 5(b) $n = 40$,用向前欧拉公式和一、四阶泰勒逼近法及 double 求初值问题的图

由表 10 - 3 和图 10 - 5(a)及图 10 - 5(b)可见,取 $n = 40$ 时,用向前欧拉公式和四阶泰勒逼近法所求初值问题的数值解的曲线几乎重合,二者的绝对误差和相对误差最小. 一阶泰勒逼近法所求初值问题的数值解的曲线与前两条曲线较近,而用函数 double 所求初值问题的曲线与前三条曲线很远,且绝对误差和相对误差很大,尤其是相对误差,当 $x = 0$ 时,Y, y_1 和 y 与真值 0 相等,而 $S_1 = 0.6912e-093$不等于真值 0,由此推测,用向前欧拉公式和四阶、一阶泰勒逼近法所求初值问题的值都接近于真值,是有实际意义的,而用函数 double 计算的值严重失真,没有实际意义.

下面再取 $n = 80$ 验证.

(4) 取 $n = 80$ 时,输入程序

```
>>syms u
 X =0:0.0250:2;
 S=exp(-sin(X))*int(exp(sin(u)),u,0,X); S1 = double(S);
 subplot(4,1,1)
```

```
        plot(X,S1,'bo'), grid
        legend('用double计算 dy/dx=1-y cos(X),y(0)=0 在[0,2]上的
精确解的数值解')
        subplot(4,1,2)
        y=exp(-sin(X)).*(1+X-cos(X)); plot(X,y,'g-'), grid
        legend('y=exp(-sin(X))(1+X-cos(X))在[0,2]上图形')
        subplot(4,1,3)
        y1=exp(-sin(X)).*( 81/64* X-10/9*cos(X)-17/64*cos
(X).*sin(X)-1/18*sin(X).^2.*cos(X)-1/96*sin(X).^3.*cos(X)+10/
9); plot(X,y1,'m-'), grid
        legend('四阶泰勒逼近的解在[0,2]上图形')
        subplot(4,1,4)
        x0=0;y0=0;b=2-1.e-4;n=80;tol=1.e-4;
        [h,k,X,Y,P,Ren]=QEuler1(@funfcn,x0,y0,b,n,tol)
        legend('n=80时,用向前欧拉公式计算 dy/dx=1-y cos(X),y(0)=0
在[0,2]上的数值解')
        syms u
        X=0:0.0250:2;
        S=exp(-sin(X))*int(exp(sin(u)),u,0,X); S1=double(S);
        jwS1y=S1'-y', xwS1y=jwS1y./S1',
        jwYy=Y-y', xwYy=jwYy./y',
        k1=1:n;k=[0,k1]; P=[k',X',Y,y',jwYy,xwYy],
        PS=[k',X',S1',jwS1y],xwS1y
```

运行后屏幕显示取 $n=80$ 时,分别用向前欧拉公式(10.8)和一、四阶泰勒多项式逼近法及其函数 double 求解此初值问题在[0,2]上的与自变量 X 对应的数值解 Y,y,y_1,S_1 及其图形(见图 10-6),Y 与 y,S_1 与 y 的绝对误差向量 $jwYy$, xwS_1y 和相对误差向量 $xwYy,xwS_1y$(略). 取 $n=80$ 时,向前欧拉公式在每个子区间$[x_k,x_{k+1}]$上的局部截断误差公式

$$Ren\approx 0.000\,3y''(\xi_k),\xi_k\in[x_k,x_{k+1}],k=1,2,\cdots,80.$$

由运行的结果可见,当 $x=0$ 时,Y,y_1和 y 与真值0相等,但是 S_1 却随着 n 的改变而发生变化,但是不等于真值0,由此推测,用向前欧拉公式和一、四阶泰勒逼近法所求初值问题的值都接近真值接近,是有实际意义的;而用函数 double 计算的值严重失真,没有实际意义. 下面再取 $n=100$ 时,向前欧拉公式在每个子区间$[x_k,x_{k+1}]$上的局部截断误差公式

$$Ren\cong 1.999\,8e-004y''(\xi_k),\xi_k\in[x_k,x_{k+1}],k=1,2,\cdots,100.$$

这说明,节点数 n 越大,向前欧拉公式在每个子区间$[x_k,x_{k+1}]$上的局部截断误差越小,所以函数 double 计算的值严重失真.

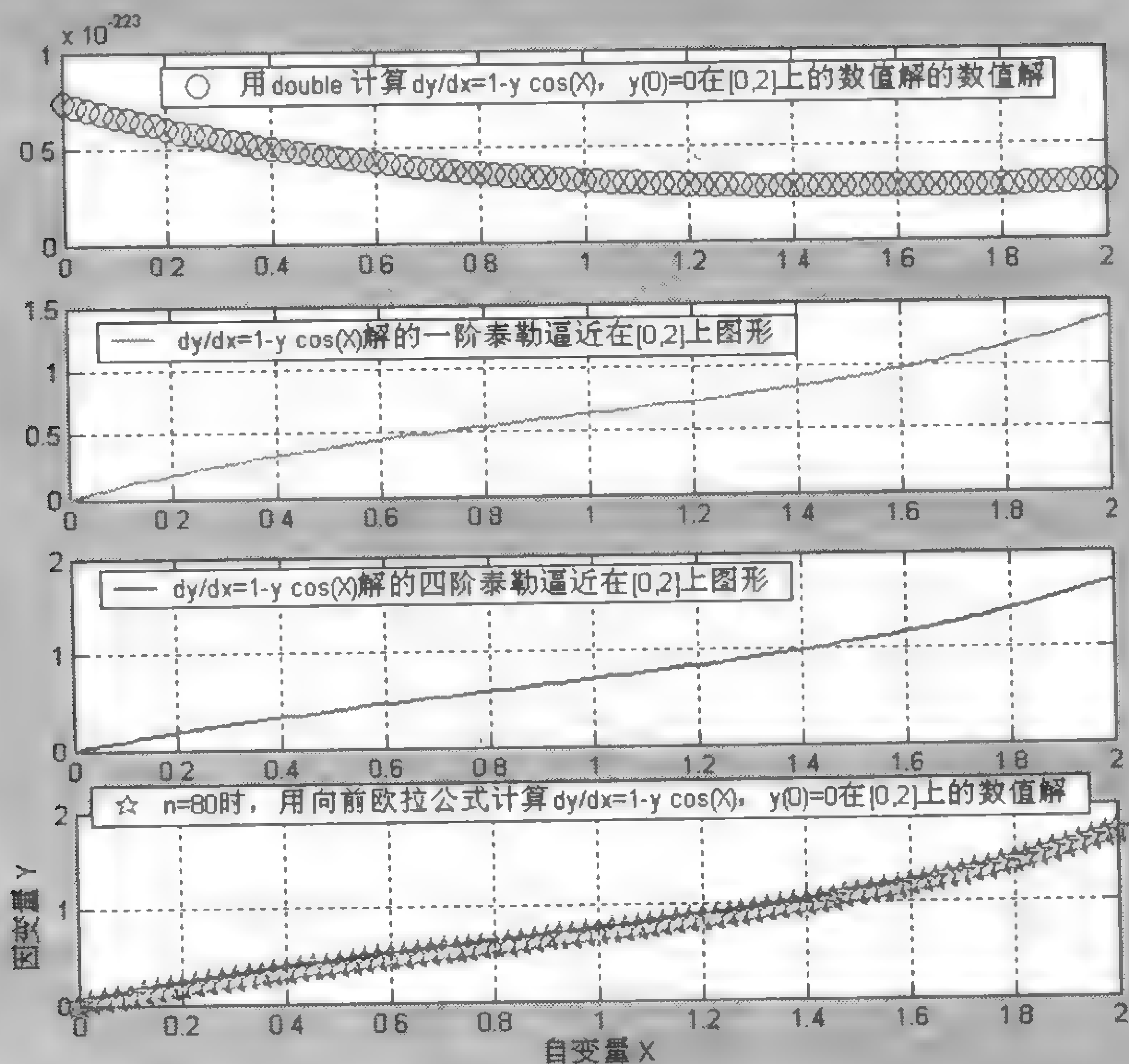


图 10-6 取 $n=80$ 时,用向前欧拉公式和一、四阶泰勒逼近法及 double 求初值问题的图

(二) 向前欧拉公式的 MATLAB 主程序 2

下面介绍根据向前欧拉公式(10.8)编写的数值计算常微分方程初值问题(10.5)的另外一种 MATLAB 程序.

> **用向前欧拉公式(10.8)求解常微分方程初值问题(10.5)的数值解及其截断误差公式的 MATLAB 主程序 2**
>
> 输入量:*funfcn* 是函数 $f(x,y)$,x_0 和 y_0 是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,h 是步长.
>
> 输出量:k 是自变量 x 取值的个数,向量 X 的元素是自变量 x 的取值,向量 Y 的元素是利用向前欧拉公式(10.8)求出常微分方程初值问题(10.5)在向量 X 的元素处的数值解,*REn* 是在每个子区间 $[x_k,x_{k+1}]$ 上的局部截断误差公式,其中 dy2 是 $y(x)$ 的 2 阶导数 $y''(x)$.

```
function [k,X,Y,P,REn] =Qeuler2(funfcn,x0,y0,b,h)
x =x0; n =fix((b -x)/h); X =zeros(n +1,1); y =y0;
Y =zeros(n +1,1); k =1; X(k) =x; Y(k) =y';
for k =2:n +1
X(k) =x +(k -1) * h,fxy =feval(funfcn,x,y), Y(k) =y +h * fxy
y =Y(k), k =k +1,plot(X,Y,'rp'),
grid,xlabel('自变量 X'), ylabel('因变量 Y')
title('用向前欧拉公式计算 dy/dx =f(x,y),y(x0) =y0 在[x0,b]上的数值解')
end
k1 =1:n;k =[0,k1]; P =[k',X,Y];
syms dy2,REn =0.5 * dy2 * h^2;
```

例 10.3.4 用向前欧拉公式(10.8)求解初值问题$\frac{dy}{dx}=y-\frac{x}{3y}, y(0)=1$,取$n=10,100$,并将计算结果与精确解作比较,写出在每个子区间$[x_k,x_{k+1}]$上的局部截断误差公式,画出数值解与精确解在区间[0,2]上的图形.

解 (1) 建立并保存名为 funfcn.m 的 M 文件函数

```
function f = funfcn (x,y)
      f =y -x/(3 *y);
```

(2) 建立并保存名为 Qeuler2.m 的 M 文件函数.

(3) 输入程序

```
>>S1 =dsolve('Dy =y -x/(3 *y)','y(0) =1','x')
```

运行后屏幕显示结果

```
S1 =
   1/6 *(6 +12 *x +30 *exp(2 *x))^(1/2)
```

(4) 输入程序

```
>> subplot(2,1,1)
x0 =0;y0 =1;b =2;n =10;h =2/10;
[k,X,Y,P,REn] =Qeuler2(@funfcn,x0,y0,b,h)
hold on
S1 =1/6 *(6 +12 *X +30 *exp(2 *X)).^(1/2); plot(X,S1,'b -')
title('用向前欧拉公式计算 dy/dx =y -x/(3y),y(0) =1 在[0,2]上的数值解')
legend('n =10 时,dy/dx =y -x/(3y),y(0) =1 在[0,2]上的数值解','dy/dx =y -x/(3y),y(0) =1 在[0,2]上的精确解')
hold off
jdwucY =S1 -Y; jwY =S1 -Y;xwY =jwY./Y;k1 =1:n;k =[0,k1];
P1 =[k',X,Y,S1,jwY,xwY]
```

```
        subplot(2,1,2)
        n1 =100;h1 =2/100;
        [k,X1,Y1,P1,Ren1] =Qeuler2(@funfcn,x0,y0,b,h1)
        hold on
        S2 =1/6*(6+12*X1+30*exp(2*X1)).^(1/2); plot(X1,S2,'b-')
        legend('n=100时,dy/dx=y-x/(3y),y(0)=1在[0,2]上的数值解','
dy/dx =y-x/(3y),y(0)=1在[0,2]上的精确解')
        hold off
        jwY1 =S2 -Y1;xwY1 =jwY1./Y1;k1 =1:n1;k =[0,k1];
        P2 =[k',X1,Y1,S2,jwY1,xwY1]
```

运行后屏幕显示取 $n=10,100$ 时,此初值问题在[0,2]上的自变量 x 的向量 $\boldsymbol{X}$,$\boldsymbol{X}_1$,利用向前欧拉公式(10.8)求出的与 $\boldsymbol{X},\boldsymbol{X}_1$ 对应的数值解 $\boldsymbol{Y},\boldsymbol{Y}_1$ 及其绝对误差向量 jwY,jwY_1 和相对误差向量 xwY,xwY_1(略),每个子区间 $[x_k,x_{k+1}]$ 上的局部截断误差公式 $Ren=\frac{1}{50}dy2$,数值解 $\boldsymbol{Y}$ 与精确解的图形(见图10-7).

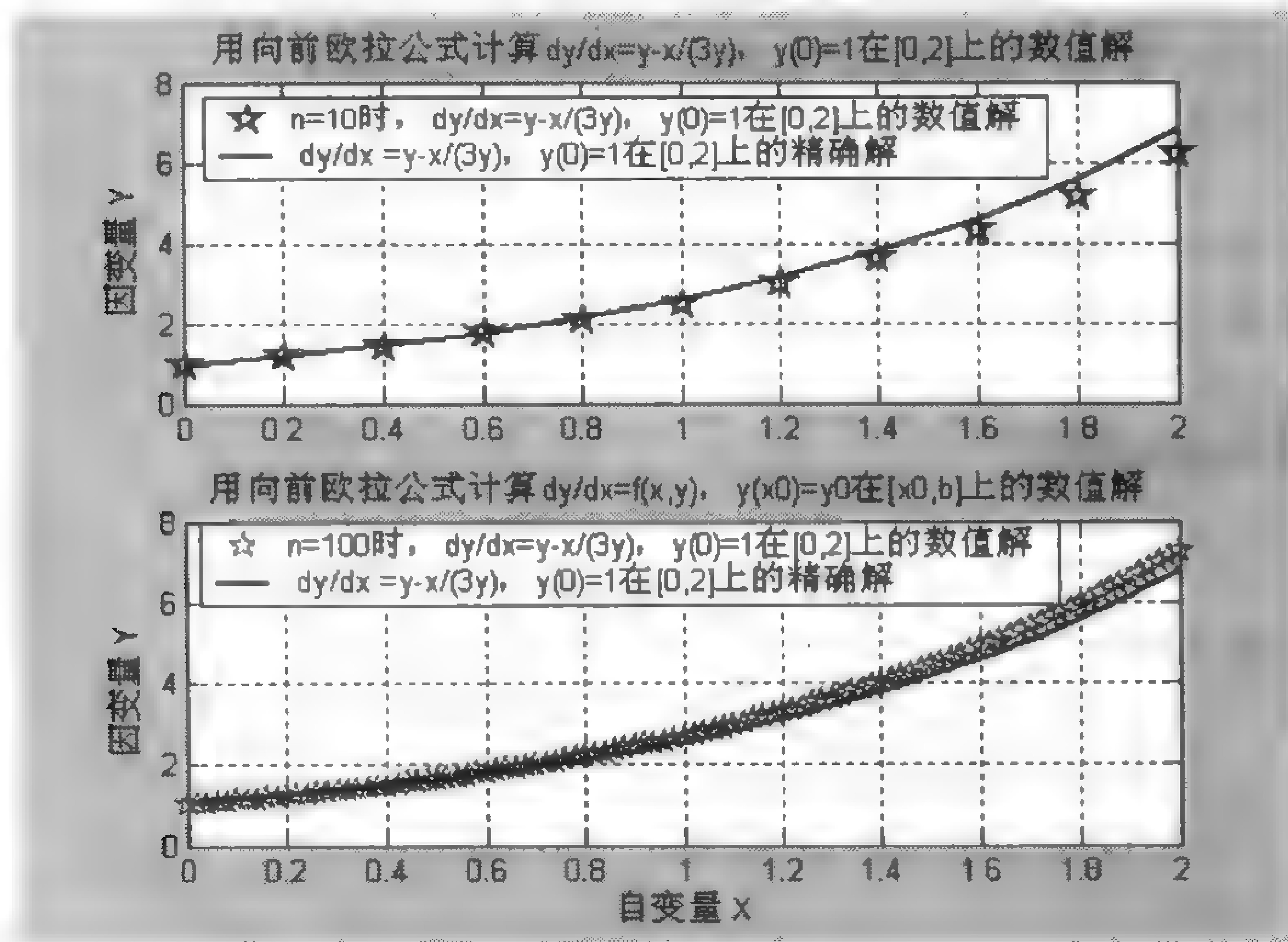


图10-7 取 $n=10,100$ 时,用向前欧拉公式求初值问题数值解和精确解的图形

(三) 自适应向前欧拉公式的 MATLAB 主程序

根据自适应方法和向前欧拉公式(10.8)编写的数值计算常微分方程初值问题(10.5)的另一个 MATLAB 程序.

用自适应向前欧拉公式求解常微分方程初值问题(10.5)的数值解的MATLAB 主程序 2

输入量:*funfcn* 是函数 $f(x,y)$,x_0 和 y_0 是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,*tol* 是精度.

输出量:向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值,向量 $\boldsymbol{Y}$ 的元素是利用向前欧拉公式(10.8)和自适应方法求出常微分方程初值问题(10.5)在向量 $\boldsymbol{X}$ 的元素处的数值解,向量 $\boldsymbol{H}$ 是步长 h 序列,n 是自变量 x 取值的序号. 并画出数值解向量 $\boldsymbol{Y}$ 的图形.

```
function [H,X,Y,k,h,P] = QEuler(funfcn,x0,b,y0,tol)
% 初始化.
pow = 1/3; if nargin < 5 | isempty(tol), tol = 1.e-6; end;
if nargin < 6 | isempty(trace), trace = 0; end;
x = x0; h = 0.0078125 * (b - x); y = y0(:);p = 128; H = zeros(p,1);
X = zeros(p,1); Y = zeros(p,length(y)); k = 1; X(k) = x; Y(k,:) = y';
% 绘图.
clc,x,h,y
end
% 主循环.
while (x < b)&(x + h > x)
  if x + h > b
h - b - x;
   end
    % 计算斜率.
    fxy = feval(funfcn,x,y); fxy = fxy(:);
% 计算误差,设定可接受误差.
  delta = norm(h * fxy,' inf '); wucha = tol * max(norm(y,' inf '),1.0);
  % 当误差可接受时重写解.
  if delta < = wucha
      x = x + h; y = y + h * fxy; k = k + 1;
    if k > length(X)
      X = [X;zeros(p,1)]; Y = [Y;zeros(p,length(y))];
      H = [H;zeros(p,1)];
    end
    H(k) = h;X(k) = x;Y(k,:) = y'; plot(X,Y,'rp'), grid
    xlabel('自变量 X'), ylabel('因变量 Y')
    title('用向前欧拉公式计算 dy/dx = f(x,y),y(x0) = y0 在[x0,b]上
```

```
的数值解')
        end
        % 更新步长.
        if delta ~ =0.0
            h = min(h*8,0.9*h*(wucha/delta)^pow);
          end
        end
        if (x<b)
        disp('Singularity likely.'), x
        end
        H = H(1:k);  X = X(1:k); Y = Y(1:k,:); n = 1:k; P = [n',H,X,Y]
```

例 10.3.5 用向前欧拉公式(10.8)求解在区间[0,2]上的初值问题$\frac{dy}{dx}=-3y+8x-7,y(0)=1$,取精度为$10^{-1}$并与精确解作比较,并在同一个坐标系中作出图形.

解 (1) 建立并保存以 funfcn.m 文件命名的 M 文件函数

```
function f = funfcn (x,y)
        f =8*x-3*y-7;
```

(2) 建立并保存以 QEuler.m 文件命名的 M 文件函数.

(3) 输入程序

```
>>S1 =dsolve('Dy =8*x-3*y-7','y(0) =1','X')
>> x0 =0;y0 =1; b =2;tol =1.e-1;
[H,X,Y,k,h,P] =QEuler(@funfcn,x0,b,y0,tol)
hold on
S1 = 8/3*X-29/9+38/9*exp(-3*X), plot(X,S1,'b-')
legend('用向前欧拉公式计算 dy/dx =8x-3y-7,y(0) =1 在[0,2]上的数
值解','dy/dx =8x-3y-7,y(0) =1 在[0,2]上的精确解')
hold off,
juwY =S1-Y; xiwY =juwY./Y; L =[P,S1,juwY,xiwY]
```

运行后屏幕显示用向前欧拉公式在[0,2]上的自变量 X 处数值解 Y 和精确解 S_1 及其图形(见图 10-8),步长 H,Y 的相对误差 $xiwY$ 和绝对误差 $juwY$ 见表 10-4.

表 10-4 取精度为 10^{-1} 时,用向前欧拉公式求初值问题的结果

n	H	X	Y	S_1	$juwY$	$xiwY$
1	0	0	1.000 0	1.000 0	0	0
2	0.009 1	0.009 1	0.908 6	0.910 2	0.001 6	0.001 7
3	0.008 5	0.017 6	0.826 8	0.829 7	0.002 8	0.003 4

续表

n	H	X	Y	S_1	$juwY$	$xiwY$
4	0.008 2	0.025 8	0.750 7	0.754 6	0.004 0	0.005 3
5	0.008 0	0.033 8	0.678 0	0.682 9	0.005 0	0.007 4
6	0.008 0	0.041 8	0.607 5	0.613 4	0.006 0	0.009 8
⋮	⋮	⋮	⋮	⋮	⋮	⋮
42	0.039 3	1.031 3	-0.323 8	-0.280 6	0.043 1	-0.133 3
43	0.037 4	1.068 7	-0.240 7	-0.201 3	0.039 5	-0.163 9
44	0.035 8	1.104 5	-0.159 4	-0.123 3	0.036 2	-0.226 9
45	0.034 5	1.139 0	-0.079 6	-0.046 3	0.033 2	-0.417 6
46	0.033 5	1.172 5	-0.000 9	0.029 7	0.030 6	-34.435 8
47	0.032 6	1.205 1	0.076 9	0.105 0	0.028 2	0.366 4
48	0.031 9	1.237 0	0.153 8	0.179 8	0.026 0	0.168 8
⋮	⋮	⋮	⋮	⋮	⋮	⋮
70	0.042 6	1.934 2	1.944 3	1.948 5	0.004 2	0.002 2
71	0.045 2	1.979 4	2.063 5	2.067 3	0.003 7	0.001 8
72	0.020 6	2.000 0	2.118 1	2.121 6	0.003 5	0.001 7

从图 10-8 和表 10-4 可以看出，取精度为 10^{-1}时，用向前欧拉公式求初值问题的数值解和精确值的绝对误差和相对误差除个别点外，总体来说达到了要求的精度 10^{-1}，说明此方法是很有效的.

10.3.3 向后欧拉公式及其误差分析

如果在用差商$\dfrac{y(x_{n+1})-y(x_n)}{h}$代替(10.5)方程的导数 y'后，$f(x,y)$中的 x 取小区间 $[x_n,x_{n+1}]$ 的右端点 x_{n+1}，再同样地以 y_n, y_{n+1} 分别代替 $y(x_n)$，$y(x_{n+1})$，就得到

$$y_{n+1}=y_n+hf(x_{n+1},y_{n+1}),n=0,1,2,\cdots \tag{10.15}$$

称为**向后欧拉公式**，它也可看作(10.9)式中积分用右端点矩形公式近似的结果. 但是(10.15)式右端的 y_{n+1}未知，所以称**隐式公式**，无法用它直接计算 y_{n+1}.

隐式公式(10.15)通常用迭代法计算，即先由向前欧拉公式产生初值

$$y_{n+1}^{(0)}=y_n+hf(x_n,y_n),n=0,1,2,\cdots. \tag{10.16}$$

再按下式进行迭代

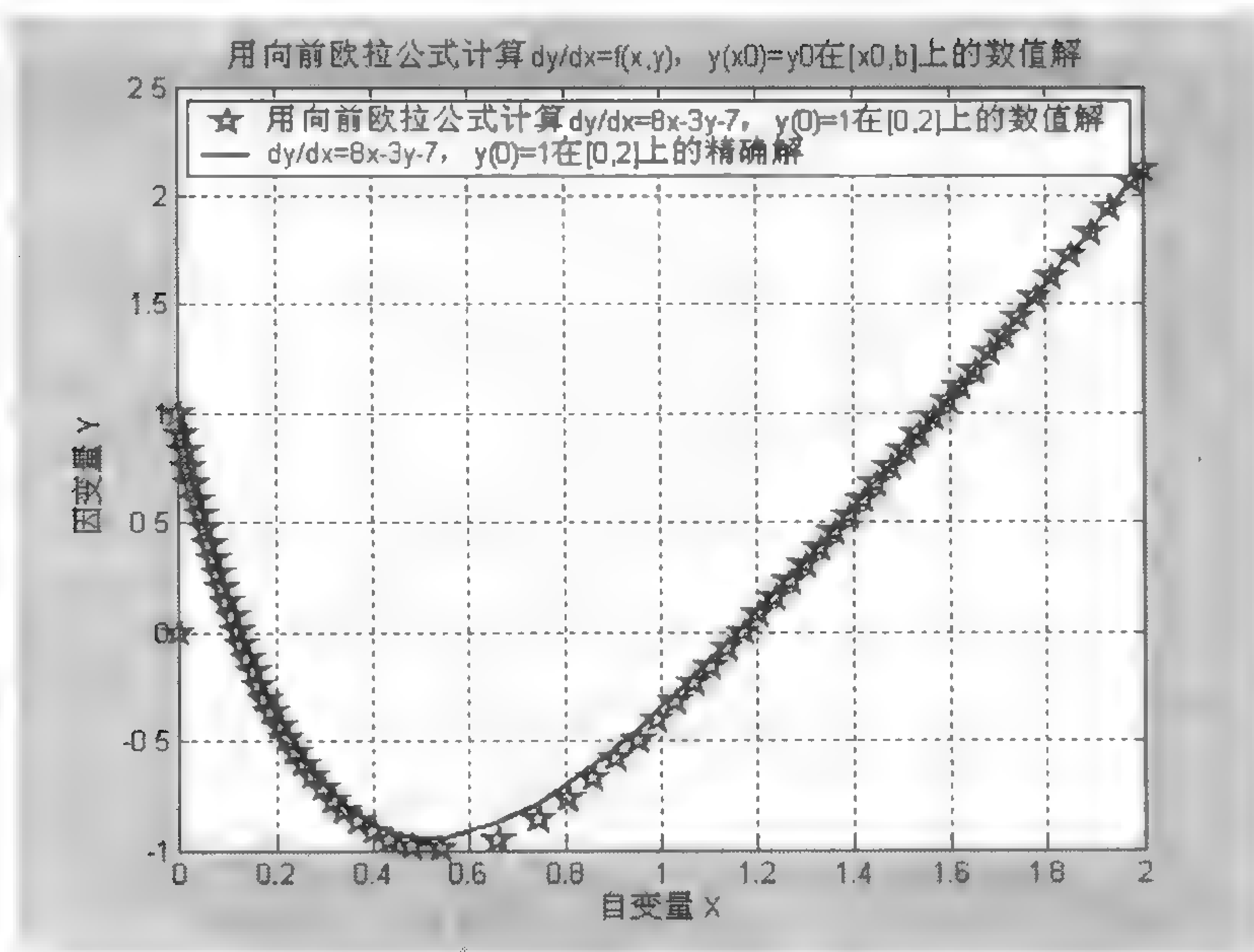


图 10-8 取精度为 10^{-1} 时,用向前欧拉公式求初值问题和精确值的图形

$$y_{n+1}^{(k+1)} = y_n + hf(x_{n+1}, y_{n+1}^{(k)}), k = 0,1,2,\cdots, n = 0,1,2,\cdots. \tag{10.17}$$

若序列 $\{y_{n+1}^{(k)}\}|_{k=0}^{+\infty}$ 收敛,则

$$\lim_{k\to+\infty} y_{n+1}^{(k)} = y_{n+1}.$$

为了估计(10.15)式的局部误差. 将右端的 $f(x,y(x))$ 在精确点 $(x_{n+1}, y(x_{n+1}))$ 展开,并注意到 $y_n = y(x_n)$,有

$$y_{n+1} = y(x_n) + h\{f(x_{n+1}, y(x_{n+1})) + f_y(x_{n+1}, \eta_{n+1})[y_{n+1} - y(x_{n+1})]\}, \tag{10.18}$$

其中 η_{n+1} 在 $y(x_{n+1})$ 与 y_{n+1} 之间. 再将

$$f(x_{n+1}, y(x_{n+1})) = y'(x_{n+1}) = y'(x_n) + hy''(x_n) + O(h^2) \tag{10.19}$$

代入(10.18)后,与(10.12)式相减,得

$$y(x_{n+1}) - y_{n+1} = -\frac{h^2}{2}y''(x_n) + O(h^3) + hf_y(x_{n+1}, \eta_{n+1})[y(x_{n+1}) - y_{n+1}], \tag{10.20}$$

于是

$$y(x_{n+1}) - y_{n+1} = \frac{-\frac{h^2}{2}y''(x_n) + O(h^3)}{1 - hf_y(x_{n+1}, \eta_{n+1})} = -\frac{h^2}{2}y''(x_n) + O(h^3) \cong O(h^2). \tag{10.21}$$

与向前欧拉公式的误差估计(10.13)式相比,我们注意到它们的 h^2 阶项数值相等,而符号相反,但二者都只具有 1 阶精度.

将向前欧拉公式(10.7)及其误差估计(10.13)式,和向后欧拉公式(10.15)及其误差估计(10.21)式的几何解释作一对比,是十分有益的.

为讨论局部截断误差,在图 10-9 中设点 $P_n(x_n,y_n)$ 落在积分曲线 $y=y(x)$ 上.按照(10.7)式,过点 P_n 以斜率 $f(x_n,y_n)$ 作直线,在 $x=x_{n+1}$ 处得到点 $A(x_{n+1},y_{n+1}^{(1)})$;而按照(10.15)式,过点 P_n 应以点 $Q(x_{n+1},y_{n+1})$ 的斜率 $f(x_{n+1},y_{n+1})$ 作直线,在 $x=x_{n+1}$ 处得到点 $B(x_{n+1},y_{n+1}^{(2)})$.这里 $y_{n+1}^{(1)},y_{n+1}^{(2)}$ 就分别是由向前欧拉公式和向后欧拉公式一步计算的结果(假定 $y_n=y(x_n)$).可以看出,A,B 一定分别在 Q 点的上、下两边.由此你会想到,如果把二者平均一下,一定能得到更好的结果.

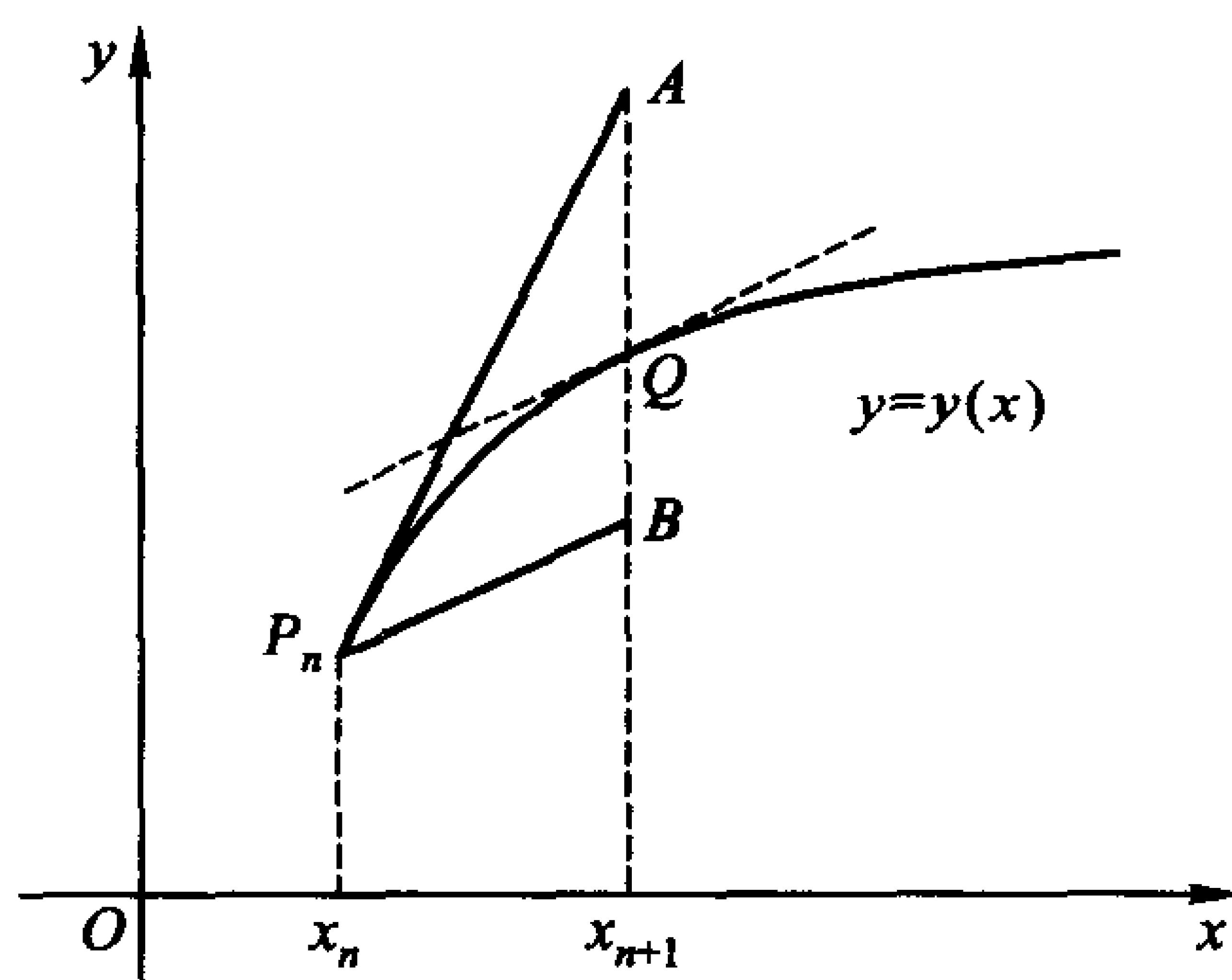


图 10-9 向前和向后欧拉公式的"平均"

请读者结合图 10-9 对(10.13)式和(10.21)式的右端 h^2 阶项的符号相反给出解释.

10.3.4 向后欧拉方法的 MATLAB 程序

根据先由向前欧拉公式(10.16)

$$y_{n+1}^{(0)}=y_n+hf(x_n,y_n),n=0,1,2,\cdots$$

产生初值,再按向后欧拉公式(10.17)式

$$y_{n+1}^{(k+1)}=y_n+hf(x_{n+1},y_{n+1}^{(k)}),k=0,1,2,\cdots,n=0,1,2,\cdots$$

进行迭代的方法,现提供求解常微分方程初值问题(10.5)的 MATLAB 程序.

用向后欧拉方法求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序

输入量:*funfcn* 是函数 $f(x,y)$,x_0 和 y_0 是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,h 是步长,*tol* 是精度.

输出量:向量 X 的元素是自变量 x 的取值,向量 Y 的元素是利用(10.16)式和(10.17)式求出初值问题(10.5)在向量 X 的元素处的数值解,n 是自变量 x 取值的个数,并画出数值解向量 Y 的图形.

```
function [X,Y,n,P] = Heuler1(funfcn,x0,b,y0,h,tol)
n = fix((b - x0)/h); X = zeros(n + 1,1); Y = zeros(n + 1,1);
```

```
k=1; X(k)=x0; Y(k,:)=y0; Y1(k,:)=y0;
% 绘图.
clc,x0,h,y0
% 产生初值.
for i=2:n+1
X(i)=x0+h; Y(i,:)=y0+h*feval(funfcn,x0,y0);
Y1(i,:)=y0+h*feval(funfcn,X(i),Y(i,:));
% 主循环.
Wu=abs(Y1(i,:)-Y(i,:));
 while Wu>tol
 p=Y1(i,:); Y1(i,:)=Y1(i-1,:)+h*feval(funfcn,X(i),p);
 end
x0=x0+h;plot(X,Y,'ro')
grid on
xlabel('自变量 X'), ylabel('因变量 Y')
end
  X=X(1: n+1); Y=Y(1: n+1,:); n=1: n+1; P=[n',X,Y]
```

例 10.3.6 用向后欧拉公式求解区间[0,2]上的初值问题$\frac{dy}{dx}=-3y+8x-7,y(0)=1$的数值解,取步长 $h=0.05$,并与精确解作比较,在同一个坐标系中作出图形.然后再取 $h=0.01$,观察数值解与精确解误差的变化,说明 h 与误差的关系.

解 (1) 建立并保存命名为 funfcn.m 的 M 文件函数

```
function f = funfcn (x,y)
        f =8*x-3*y-7;
```

(2) 建立并保存以 Heuler1.m 文件命名的 M 文件函数.

(3) 输入程序

```
>>S1=dsolve('Dy=8*x-3*y-7','y(0)=1','x')
>> x0=0;y0=1; b=2;tol=1.e-1;
subplot(2,1,1)
h1=0.01; [X1,Y1,n,P1]=Heuler1(@funfcn,x0,b,y0,h1,tol)
hold on
S2 = 8/3*X1-29/9+38/9*exp(-3*X1), plot(X1,S2,'b-')
legend('h=0.01 用向后欧拉公式计算 dy/dx=8x-3y-7,y(0)=1 在[0,
2]上的数值解','dy/dx=8x-3y-7,y(0)=1 在[0,2]上的精确解')
hold off
juwY1=S2-Y1; xiwY1=juwY1./Y1; L=[P1,S2,juwY1,xiwY1]
subplot(2,1,2)
```

```
h=0.05; [X,Y,n,P]=Heuler1(@funfcn,x0,b,y0,h,tol)
hold on
S1 = 8/3*X-29/9+38/9*exp(-3*X), plot(X,S1,'b-')
legend('h=0.05 用向后欧拉公式计算 dy/dx=8x-3y-7,y(0)=1 在[0,
2]上的数值解','dy/dx=8x-3y-7,y(0)=1 在[0,2]上的精确解')
hold off
juwY=S1-Y; xiwY=juwY./Y; L=[P,S1,juwY,xiwY]
```

运行后屏幕显示用向后欧拉公式计算此初值问题在[0,2]上的自变量 X 处数值解 Y 和精确解 S_1 及其图形(见图 10-10),步长 H,Y 的相对误差 *xiwY* 和绝对误差 *juwY*(见表 10-5).

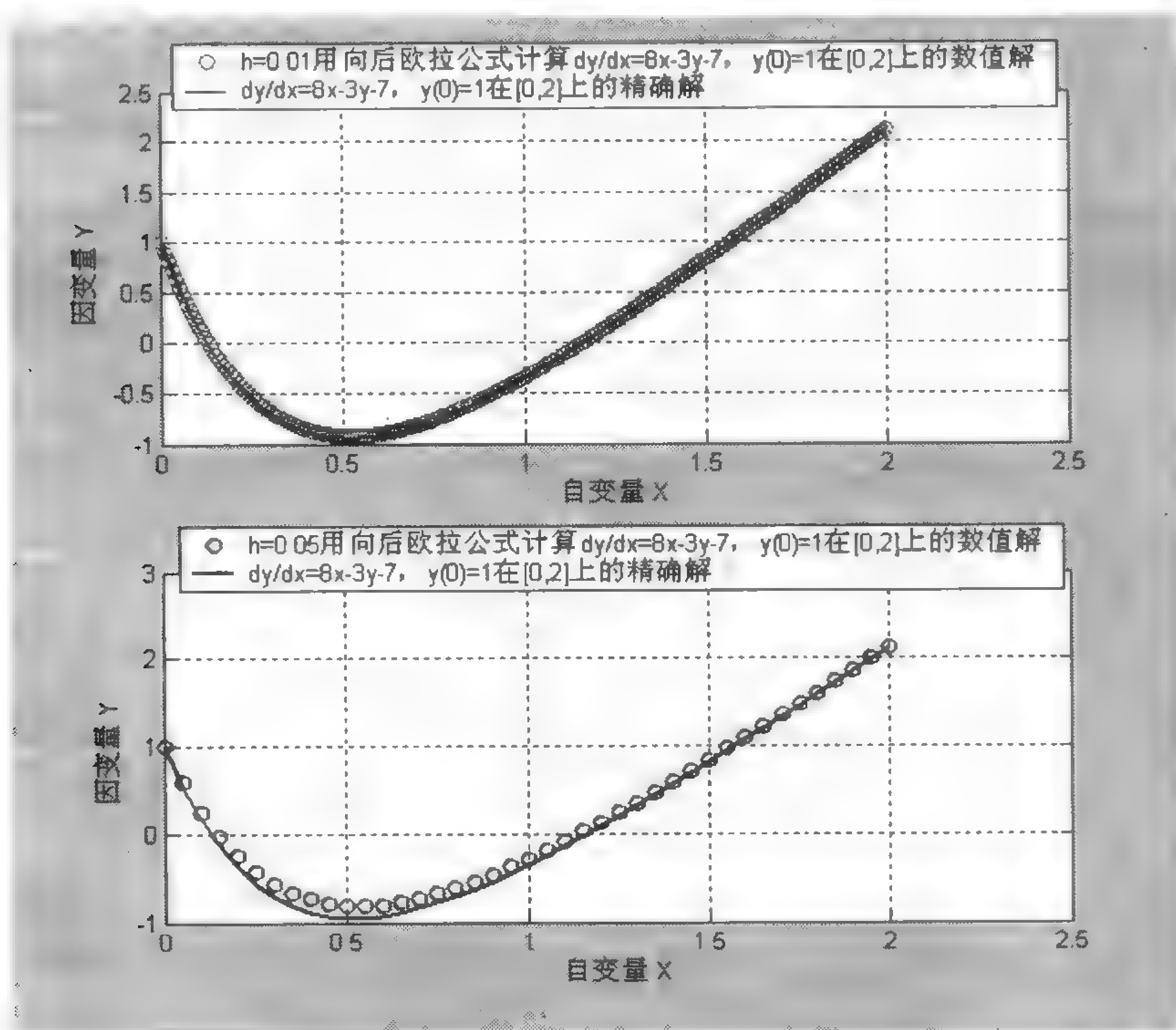


图 10-10 用向后欧拉公式求初值问题的数值解和精确解的图形

表 10-5 取 $h=0.05$ 时,用向后欧拉公式求初值问题的数值解和精确解及其误差

n	X	Y	S_1	*juwY*	*xiwY*
1	0	1.000 0	1.000 0	0	0
2	0.050 0	0.595 0	0.545 2	-0.049 8	-0.083 7

续表

n	X	Y	S_1	juwY	xiwY
3	0.100 0	0.258 6	0.172 3	-0.086 3	-0.333 6
4	0.150 0	-0.017 8	-0.130 0	-0.112 2	6.288 3
5	0.200 0	-0.242 1	-0.371 7	-0.129 6	0.535 5
6	0.250 0	-0.420 7	-0.561 1	-0.140 4	0.333 8
⋮	⋮	⋮	⋮	⋮	⋮
23	1.100 0	-0.078 8	-0.133 2	-0.054 3	0.689 6
24	1.150 0	0.027 7	-0.021 5	-0.049 3	-1.775 8
⋮	⋮	⋮	⋮	⋮	⋮
40	1.950 0	1.998 5	1.989 9	-0.008 5	-0.004 3
41	2.000 0	2.129 1	2.121 6	-0.007 6	-0.003 6

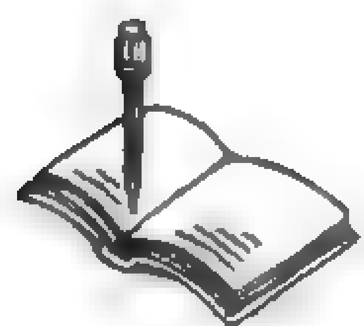
表 10-6 取 $h=0.01$ 时,用向后欧拉公式求初值问题的数值解和精确解及其误差

n	X	Y	S_1	juwY	xiwY
1	0	1.000 0	1.000 0	0	0
2	0.010 0	0.903 8	0.901 9	-0.001 9	-0.002 1
3	0.020 0	0.811 2	0.807 5	-0.003 7	-0.004 6
⋮	⋮	⋮	⋮	⋮	⋮
11	0.100 0	0.187 0	0.172 3	-0.014 7	-0.078 5
12	0.110 0	0.122 2	0.106 6	-0.015 7	-0.128 2
13	0.120 0	0.060 1	0.043 5	-0.016 6	-0.276 1
14	0.130 0	0.000 6	-0.016 9	-0.017 5	-30.005 0
15	0.140 0	-0.056 4	-0.074 7	-0.018 2	0.323 2
⋮	⋮	⋮	⋮	⋮	⋮
112	1.110 0	-0.103 0	-0.111 1	-0.008 1	0.078 2
113	1.120 0	-0.081 0	-0.088 9	-0.007 9	0.097 5
114	1.130 0	-0.058 8	-0.066 6	-0.007 7	0.131 4
115	1.140 0	-0.036 5	-0.044 1	-0.007 6	0.207 3
116	1.150 0	-0.014 1	-0.021 5	-0.007 4	0.525 7
117	1.160 0	0.008 4	0.001 2	-0.007 3	-0.859 5
118	1.170 0	0.031 1	0.024 0	-0.007 1	-0.228 4
119	1.180 0	0.053 9	0.046 9	-0.007 0	-0.129 1
120	1.190 0	0.076 8	0.070 0	-0.006 8	-0.088 7
121	1.200 0	0.099 8	0.093 1	-0.006 7	-0.066 8

续表

n	X	Y	S_1	$juwY$	$xiwY$
122	1.210 0	0.122 9	0.116 4	-0.006 5	-0.053 1
123	1.220 0	0.146 1	0.139 8	-0.006 4	-0.043 7
⋮	⋮	⋮	⋮	⋮	⋮
199	1.980 0	2.070 0	2.068 9	-0.001 1	-0.000 5
200	1.990 0	2.096 3	2.095 2	-0.001 1	-0.000 5
201	2.000 0	2.122 6	2.121 6	-0.001 0	-0.000 5

由表 10-5 和图 10-10 可以看出，取 $h=0.05$ 时，除自变量 $X=0.150\,0$，$0.200\,0$，$1.100\,0$，$1.150\,0$，数值解 Y 与精确解 S_1 的相对误差 $xiwY=6.288\,3$，$0.535\,5$，$0.689\,6$，$-1.775\,8$ 外，其余的 $xiwY<0.333\,8$，尤其，当 $1.300\,0\leqslant X\leqslant 2$ 时，$-0.099\,1<xiwY<-0.003\,6$，绝对误差 $juwY$ 在 $-0.036\,3<juwY<-0.007\,6$；当 $0<X\leqslant 2$ 时，$-0.147\,7<juwY<-0.007\,6$；且满足初始条件 $Y(0)=1$. 由表 10-6 和图 10-10（上图）可以看出，取 $h=0.01$ 时，数值解与精确解的图形几乎完全重合，除 $X=0.130\,0$，$1.160\,0$，$xiwY=-30.005\,0$，$-0.859\,5$ 以外，其余的绝对误差 $juwY$ 和相对误差 $xiwY$ 比取 $h=0.05$ 时的误差明显变小，这说明总体上 h 与误差是成正比例的关系，即步长 h 越小，误差越小



习题 10.3

1. 分别用向前欧拉方法和向后欧拉方法求初值问题

$$\begin{cases}\dfrac{dy}{dx}=1-y+x,0\leqslant x\leqslant 1,\\ y|_{x=0}=1\end{cases}$$

的数值解，分别取 $h=0.075$，$0.007\,5$，并计算误差，画出精确解和数值解的图形.

2. 用向前欧拉公式(10.8)求解初值问题 $\dfrac{dy}{dx}=1+y^2+x^2$，$y(0)=0$，$0\leqslant x\leqslant 1$，取 $n=10$，50，并将计算结果与精确解作比较，写出在每个子区间 $[x_k,x_{k+1}]$ 上的局部截断误差公式，画出数值解与精确解在区间 $[0,1]$ 上的图形.

3. 用向后欧拉公式求 $y(x)=\int_0^x e^{-t^2}dt$ 在 $x=0.5,1.0,1.5,2.0$ 处的近似值.

4. 分别用向前欧拉公式和一阶泰勒多项式逼近法及其函数 double 求解初值问题

$$\begin{cases}\dfrac{dy}{dx}=1-y\cos x,0\leqslant x\leqslant 1,\\ y(0)=0.\end{cases}$$

取 $n=40,80$ 时,并将计算结果与精确解作比较,写出在每个子区间 $[x_k,x_{k+1}]$ 上的局部截断误差公式,画出数值解与精确解在区间 $[0,1]$ 上的图形,说明用函数 double 计算的结果是否有使用价值.

5. 用自适应向前欧拉方法求解区间 $[0,2]$ 上的初值问题 $\frac{dy}{dx}=-y+x^2+x, y(0)=0$,并与精确解作比较,取精度为 10^{-1},在同一个坐标系中作出图形.

10.4　改进的欧拉方法及其 MATLAB 程序

下面主要介绍梯形公式、改进的欧拉方法和它们的误差估计及其 MATLAB 程序,并通过例题说明它们的具体应用.

10.4.1　梯形公式及其误差估计

将向前欧拉公式(10.10)和向后欧拉公式和(10.15)加以平均,得到

$$y_{n+1}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_{n+1},y_{n+1})], n=0,1,2,\cdots. \tag{10.22}$$

它相当于(10.9)式:$y(x_{n+1})-y(x_n)=\int_{x_n}^{x_{n+1}} f[x,y(x)]dx (n=0,1,2,\cdots)$ 右端的积分用求面积的梯形公式近似,再以 y_n, y_{n+1} 分别代替 $y(x_n), y(x_{n+1})$ 的结果,(10.22)式称**梯形公式**.它也要用迭代方法求解,即由(10.16)式 $y_{n+1}^{(0)}=y_n+hf(x_n,y_n), n=0,1,2,\cdots$ 得到初值后按照下式进行迭代

$$y_{n+1}^{(k+1)}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_{n+1},y_{n+1}^{(k)})], \quad k=0,1,2,\cdots, n=0,1,2,\cdots \tag{10.23}$$

因为(10.13)式

$$y(x_{n+1})-y_{n+1}=\frac{h^2}{2}y''(x_n)+O(h^3)$$

和(10.21)式

$$y(x_{n+1})-y_{n+1}=-\frac{h^2}{2}y''(x_n)+O(h^3)$$

的右端 h^2 阶项刚好抵消,所以**梯形公式的局部截断误差为 $O(h^3)$,具有2阶精度**.

10.4.2　梯形公式的两种 MATLAB 程序

根据(10.16)式 $y_{n+1}^{(0)}=y_n+hf(x_n,y_n), n=0,1,2,\cdots$ 和(10.23)式

$$y_{n+1}^{(k+1)}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_{n+1},y_{n+1}^{(k)})], k=0,1,2,\cdots, n=0,1,2,\cdots$$

编写求解常微分方程初值问题(10.5)的 MATLAB 程序 1 和自适应 MATLAB 程

序 2.

(一) 梯形公式的 MATLAB 程序

用梯形公式求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序 1

输入量:*funfcn* 是函数 $f(x,y)$,x_0和 y_0是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,h 是步长,*tol* 是精度.

输出量:向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值,向量 $\boldsymbol{Y}$ 的元素是利用(10.16)式、(10.23)式和自适应方法求初值问题(10.5)在向量 $\boldsymbol{X}$ 的元素处的数值解,向量 $\boldsymbol{H}$ 是步长 h 序列,n 是自变量 x 取值的序号,并画出数值解向量 $\boldsymbol{Y}$ 的图形.

```
function [X,Y,n,P] = odtixing1(funfcn,x0,b,y0,h,tol)
n = fix((b - x0)/h); X = zeros(n +1,1); Y = zeros(n +1,1);
k =1; X(k) =x0; Y(k,:) =y0; Y1(k,:) =y0;
% 绘图.
clc,x0,h,y0
% 产生初值.
for i =2: n +1
X(i) =x0 +h; fx0y0 = feval(funfcn,x0,y0); Y(i,:) =y0 +h * fx0y0;
fxiyi = feval(funfcn,X(i),Y(i,:));
Y1(i,:) =y0 +h * (fxiyi + fx0y0)/2;
% 主循环.
Wu - abs(Y1(i,:)  Y(i,:));
    while Wu > tol
      p = Y1(i,:), fxip = feval(funfcn,X(i),p);
      Y1(i,:) =y0 +h * (fx0y0 +fxip)/2,P1 = Y1(i,:), Y(i,:) =p1;
    end
x0 =x0 +h; y0 =Y1(i,:); Y(i,:) =y0; plot(X,Y,'ro')
grid on
xlabel('自变量 X'), ylabel('因变量 Y')
title('用梯形公式计算 dy/dx = f(x,y),y(x0) =y0 在[x0,b]上的数值解')
end
 X =X(1: n +1); Y =Y(1: n +1,:); n =1: n +1; P =[n',X,Y]
```

例 10.4.1 用梯形公式求解区间[0,2]上的初值问题$\frac{dy}{dx}=-3y+8x-7$,$y(0)=1$,取步长 $h=0.05$,精度为 10^{-1},并与精确解作比较,在同一个坐标系中画出图形.

解 (1) 建立并保存名为 funfcn.m 的 M 文件函数

```
function f = funfcn (x,y)
        f =8*x-3*y-7;
```

(2) 建立并保存名为 odtixing.m 的 M 文件函数.

(3) 输入程序

```
>> S1 =dsolve('Dy =8*x-3*y-7','y(0) =1','x')
>> x0 =0;y0 =1; b =2; tol =0.1; h =0.05;
[X,Yt,n,Pt] =odtixing1(@funfcn,x0,b,y0,h,tol)
hold on
S1 =8/3*X-29/9+38/9*exp(-3*X);
plot(X,S1,'b-'), hold off
legend('h =0.05,用梯形公式计算 dy/dx =8x-3y-7,y(0) =1 在[0,2]
上的数值解','dy/dx =8x-3y-7,y(0) =1 在[0,2]上的精确解')
juwYt =S1-Yt; xiwYt =juwYt./Yt; Lt =[Pt,S1,juwYt,xiwYt]
```

运行后屏幕显示取精度为 10^{-1},用向前欧拉公式求解此初值问题在区间[0,2]上的自变量 X 处数值解 Y_t 和精确解 S_1,步长 h,Y_t 的相对误差 $xiwY_t$ 和绝对误差 $juwY_t$(略)及其数值解和精确解的图形(见图 10-11).

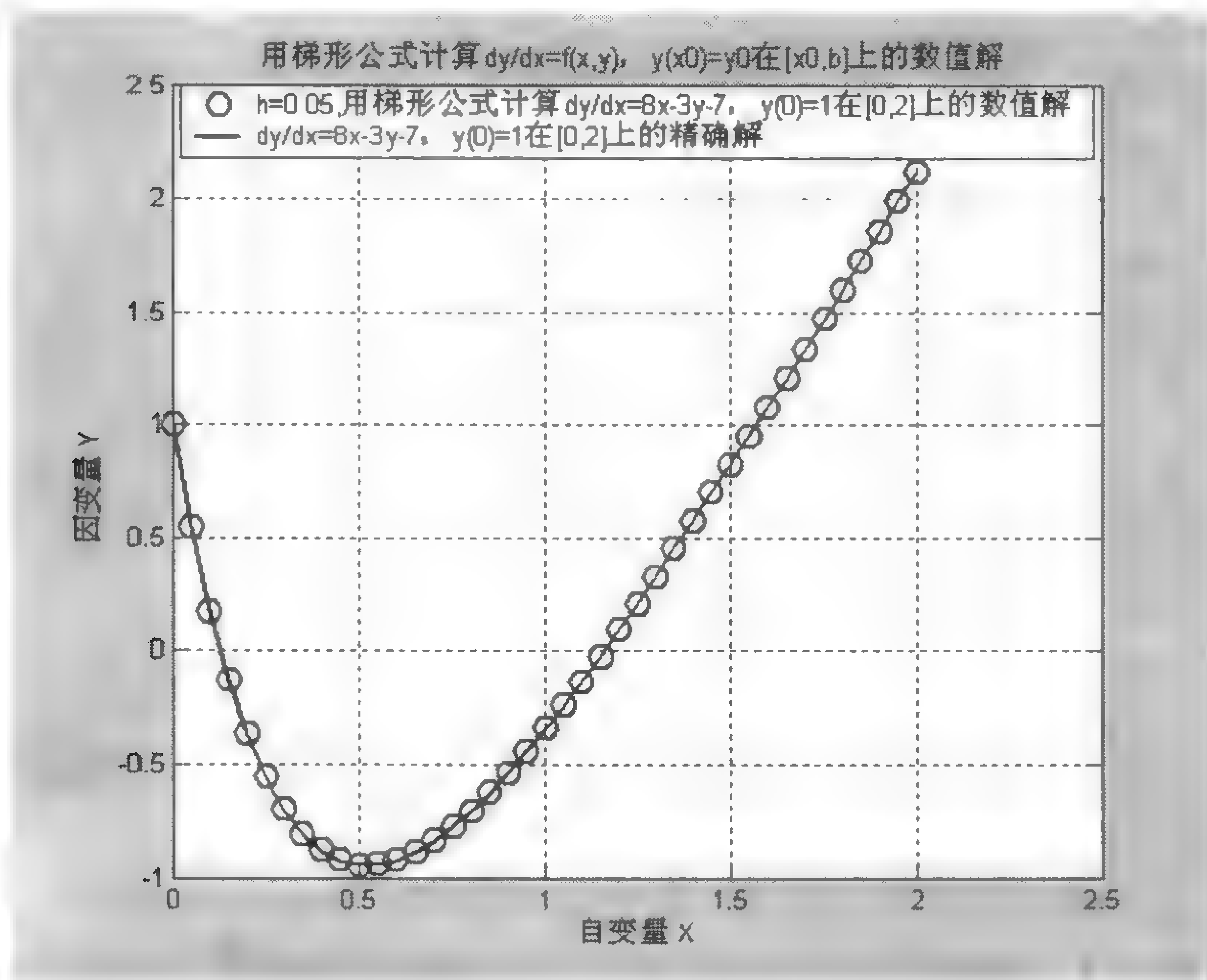


图 10-11 用梯形公式求解区间[0,2]上的初值问题 $\frac{dy}{dx}=-3y+8x-7,y(0)=1$

(二) 自适应梯形公式的 MATLAB 程序

求解常微分方程初值问题(10.5)的自适应 MATLAB 程序 2 如下：

用自适应梯形公式求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序

输入量：*funfcn* 是函数 $f(x,y)$，x_0 和 y_0 是初值 $y(x_0)=y_0$，，b 是自变量 x 的最大值，*tol* 是精度.

输出量：向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值，向量 $\boldsymbol{Y}$ 的元素是利用(10.16)式、(10.23)式和自适应方法求出初值问题(10.5)在向量 $\boldsymbol{X}$ 的元素处的数值解，向量 $\boldsymbol{H}$ 是步长 h 的序列，n 是自变量 x 取值的序号. 并画出数值解向量 $\boldsymbol{Y}$ 的图形.

```
function [H,X,Y,k,h,P] = odtixing2(funfcn,x0,b,y0,tol)
% 初始化.
pow = 1/3;
if nargin < 5 | isempty(tol),
tol = 1.e-6;
end;
if nargin < 6 | isempty(trace),
trace = 0;
end;
x = x0; h = 0.0078125 * (b - x); y = y0(:);
p = 128; n = fix((b - x0)/h); H = zeros(p,1);
X = zeros(p,1); Y = zeros(p,length(y));
k = 1; X(k) = x; Y(k,:) = y';
% 绘图.
clc,x,h,y
% 主循环.
while (x < b)&(x + h > x)
 if x + h > b
h = b - x;
   end
    % 计算斜率.
    fxy = feval(funfcn,x,y); fxy = fxy(:);
% 计算误差,设定可接受误差.
delta = norm(h * fxy,' inf ');
wucha = tol * max(norm(y,' inf '),1.0);
% 当误差可接受时重写解.
```

```
if delta < =wucha
x =x +h; y1 =y +h * fxy;
fxy1 =feval(funfcn,x,y1); fxy =fxy(:);
y2 =y +h * fxy1; y =(y1 +y2)/2; k =k +1;
  if k >length(X)
     X =[X;zeros(p,1)]; Y =[Y;zeros(p,length(y))];
     H =[H;zeros(p,1)];
  end
H(k) -h;X(k) =x;Y(k,:) =y'; plot(X,Y,'go'), grid
xlabel('自变量 X'), ylabel('因变量 Y')
title('用自适应梯形公式计算 dy/dx =f(x,y),y(x0) =y0 在[x0,b]上的解')
end
% 更新步长.
if delta ~ =0.0
h =min(h * 8,0.9 * h * (wucha/delta)^pow);
end
end
if (x <b)
disp('Singularity likely.'), x
end
H =H(1: k);  X =X(1:k); Y =Y(1: k,:); n =1: k; P =[n',H,X,Y]
```

例 10.4.2 分别用自适应梯形公式和向前欧拉公式分别求解区间[0,2]上的初值问题$\frac{dy}{dx} = -3y+8x-7, y(0)=1$,取精度为 10^{-1},并与精确解作比较,在同一个坐标系中作出图形.

解 (1) 建立并保存以 funfcn.m 文件命名的 M 文件函数

```
function f = funfcn (x,y)
        f =8 * x -3 * y -7;
```

(2) 建立并保存以 odtixing2.m 文件命名的 M 文件函数.

(3) 输入程序

```
> >S1 =dsolve('Dy =8 * x -3 * y -7','y(0) =1','X')
> > x0 =0;y0 =1; b =2; tol =1.e -1;
[Ht,X,Yt,k,h,Pt] = odtixing2(@funfcn,x0,b,y0,tol), hold on
[Hq,X,Yq,k,h,Pq] =QEuler(@funfcn,x0,b,y0,tol)
S1 = 8/3 * X -29/9 +38/9 * exp( -3 * X), plot(X,S1,'b -')
grid
legend('用自适应梯形公式计算 dy/dx =8x -3y -7,y(0) =1 在[0,2]上的
解','用向前欧拉公式计算 dy/dx =8x -3y -7,y(0) =1 在[0,2]上的数值解','dy/dx =
```

```
8x-3y-7, y(0)=1 在[0,2]上的精确解')
   hold off
```

运行后屏幕显示取精度为 10^{-1}，分别用梯形公式和向前欧拉公式求解此初值问题在区间[0,2]上的自变量 X 处数值解 $Y_i(i=t,q)$ 和精确解 S_1 及其图形（见图 10-12），步长 h 等（略）.

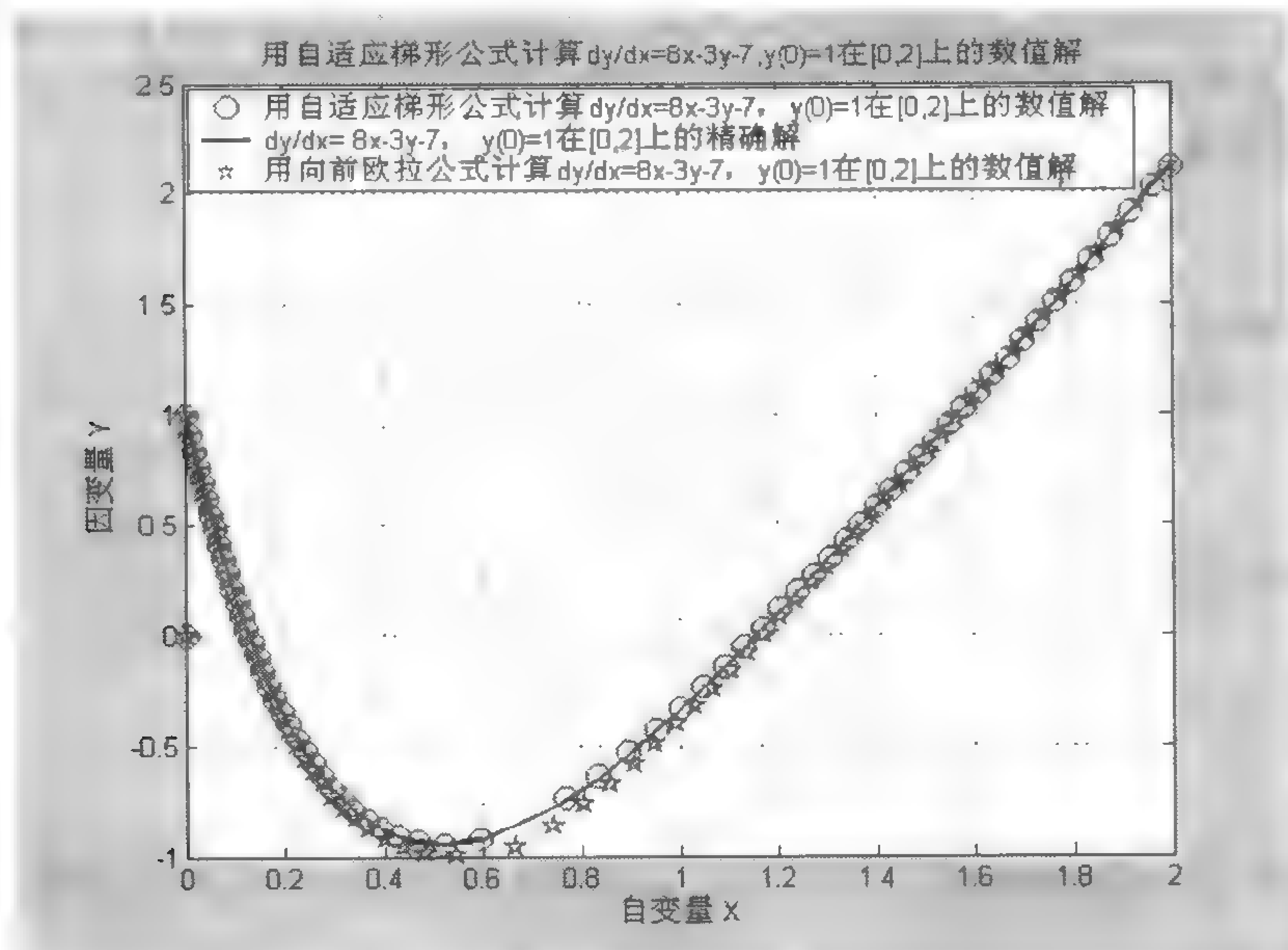


图 10-12 自适应梯形和向前欧拉公式求解初值问题 $\frac{dy}{dx}=-3y+8x-7$，$y(0)=1$ 图形

由图 10-12 可见，用自适应梯形公式求解此初值问题的数值解的图形几乎与精确解完全重合在一起，而用向前欧拉公式计算的数值解的图形与精确解在[0.4,1]比梯形公式的数值解误差大，所以，梯形公式比向前欧拉公式精度高.

10.4.3 改进的欧拉公式

我们看到，向前欧拉公式计算简单，但精度只有 1 阶，梯形公式的精度提高到 2 阶，但迭代太繁. 改进的方法是，把迭代过程简化为两步：先由向前欧拉公式(10.10)算出 y_{n+1} 的预测值 $\overline{y}_{n+1}$；再把它代入梯形公式(10.23)式右端，作为校正，即

$$\overline{y}_{n+1}=y_n+hf(x_n,y_n),$$

$$y_{n+1}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_{n+1},\overline{y}_{n+1})],n=0,1,2,\cdots \qquad (10.24)$$

称为**改进的欧拉公式**,它还可写作

$$\begin{cases} y_{n+1}=y_n+\dfrac{h}{2}(k_1+k_2), \\ k_1=f(x_n,y_n), \\ k_2=f(x_{n+1},y_n+hk_1). \end{cases} \tag{10.25}$$

对于求微分方程数值解的欧拉方法,一共介绍了 4 个公式,人们常用的是便于计算的向前欧拉公式(10.10)和改进的欧拉公式(10.25),下面看一个例子.

例 10.4.3 用向前欧拉公式和改进欧拉公式求初值问题

$$y'=y-\frac{2x}{y},y(0)=1 \tag{10.26}$$

的数值解($0\leqslant x\leqslant 1$,步长 $h=0.1$),并与精确解比较.

解 对于方程(10.26)向前欧拉公式的具体形式是

$$y_{n+1}=y_n+h\left(y_n-\frac{2x_n}{y_n}\right),n=0,1,\cdots.$$

改进欧拉公式的具体形式是

$$k_1=y_n-\frac{2x_n}{y_n},\quad k_2=y_n+hk_1-\frac{2x_{n+1}}{y_n+hk_1},$$

$$y_{n+1}=y_n+\frac{h}{2}(k_1+k_2),n=0,1,\cdots.$$

方程(10.26)的精确解是 $y=\sqrt{1+2x}$,以步长 $h=0.1$ 在 $0\leqslant x\leqslant 1$ 内的计算结果如表 10－7:

表 10－7 用向前欧拉公式和改进欧拉公式求初值问题(10.26)的数值解和精确解

x_n	向前欧拉公式	改进欧拉公式	精确解	x_n	向前欧拉公式	改进欧拉公式	精确解
	y_1	y_2	$y(x_n)$		y_1	y_2	$y(x_n)$
0.1	1.100 0	1.095 9	1.095 4	0.6	1.509 0	1.486 0	1.483 2
0.2	1.191 8	1.184 1	1.183 2	0.7	1.580 3	1.552 5	1.549 2
0.3	1.277 4	1.266 2	1.264 9	0.8	1.649 8	1.615 3	1.616 5
0.4	1.358 2	1.343 4	1.341 6	0.9	1.717 8	1.678 2	1.673 3
0.5	1.435 1	1.416 4	1.414 2	1.0	1.784 8	1.737 9	1.732 1

可以看出改进欧拉公式比向前欧拉公式的结果有很大改进,并且随着 x_n 的增加 y_n 累积误差的增长相当可观.

10.4.4 改进的欧拉公式的 MATLAB 程序

根据(10.24)式编写求解常微分方程初值问题(10.5)的 MATLAB 程序

如下.

用自适应改进的欧拉公式求解常微分方程初值问题(10.5)的数值解的MATLAB主程序

输入量:*funfcn* 是函数 $f(x,y)$,x_0 和 y_0 是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,*tol* 是精度.

输出量:向量 X 的元素是自变量 x 的取值,向量 Y 的元素是利用(10.16)式、(10.24)式和自适应方法求出初值问题(10.5)在向量 X 的元素处的数值解,向量 H 是步长 h 序列,n 是自变量 x 取值的序号. 并画出数值解向量 Y 的图形.

```
function [H,X,Y,k,h,P] = gaiEuler(funfcn,x0,b,y0,tol)
% 初始化.
pow = 1/3; if nargin < 5 | isempty(tol)
tol = 1.e-6;
end
if nargin < 6 | isempty(trace)
trace = 0
end
x = x0; h = 0.0078125 * (b - x); y = y0(:);p = 128; n = fix((b - x0)/h);
H = zeros(p,1); X = zeros(p,1);
Y = zeros(p,length(y)); k = 1; X(k) = x; Y(k,:) = y';
% 绘图.
clc,x,h,y
end
% 主循环.
while (x < b)&(x + h > x)
 if x + h > b
h = b - x;
 end
   % 计算斜率.
   fxy = feval(funfcn,x,y); fxy = fxy(:);
% 计算误差,设定可接受误差.
delta = norm(h * fxy,'inf'); wucha = tol * max(norm(y,'inf'),1.0);
% 当误差可接受时重写解.
     if delta < = wucha
         x = x + h; y1 = y + h * fxy; fxy1 = feval(funfcn,x,y1);
         fxy = fxy(:);y2 = (fxy + fxy1)/2; y = y + h * y2; k = k + 1;
```

```
            if k > length(X)
            X = [X;zeros(p,1)]; Y = [Y;zeros(p,length(y))];
            H = [H;zeros(p,1)];
            end
          H(k) = h;X(k) = x;Y(k,:) = y'; plot(X,Y,'mh'), grid
          xlabel('自变量 X'), ylabel('因变量 Y')
          title('用改进的欧拉公式计算 dy/dx = f(x,y),y(x0) = y0 在[x0,
b]上的数值解')
          end
        %更新步长.
      if delta ~ =0.0
         h = min(h * 8,0.9 * h * (wucha/delta)^pow);
       end
      end
      if (x < b)
      disp(' Singularity likely.'), x
        end
      H = H(1: k);  X = X(1: k); Y = Y(1: k,:); n = 1: k; P = [n',H,X,Y]
```

例 10.4.4 分别用梯形公式和改进的欧拉公式求解区间[0,2]上的初值问题$\frac{dy}{dx} = -3y + 8x - 7, y(0) = 1$,取精度为 10^{-1},与精确解作比较,在同一个坐标系中作出图形.

解 (1) 建立并保存以 funfcn.m 文件命名的 M 文件函数

```
function f = funfcn (x,y)
      f =8 * x - 3 * y - 7;
```

(2) 建立并保存以 gaiEuler.m 文件命名的 M 文件函数.

(3) 输入程序

```
>> S1 = dsolve(' Dy = 8 * x - 3 * y - 7 ',' y(0) = 1 ',' x ')
x0 = 0;y0 = 1; b = 2;tol = 1.e - 1;
[Ht,X,Yt,k,h,Pt] = odtixing2(@ funfcn,x0,b,y0,tol)
hold on, [H,X,Y,k,h,P] = gaiEuler(@ funfcn,x0,b,y0,tol)
S1 = 8/3 * X - 29/9 + 38/9 * exp( - 3 * X), plot(X,S1,' b - ')
legend('用梯形公式计算 dy/dx = 8x - 3y - 7,y(0) = 1 在[0,2]上的数值
解','用改进的欧拉公式计算 dy/dx = 8x - 3y - 7,y(0) = 1 在[0,2]上的数值解', ' dy/
dx = 8x  3y - 7,y(0) = 1 在[0,2]上的精确解')
hold off,juwY = S1 - Y; xiwY = juwY./Y; L = [P,S1,juwY,xiwY]
```

运行后屏幕显示取精度为 10^{-1},分别用梯形公式和改进的欧拉公式求此解初值问题在区间[0,2]上的自变量 X 处数值解 Y_t, Y 和精确解 S_1 及其图形(见

图 10－13)，步长 h，Y 的相对误差 $xiwY$ 和绝对误差 $juwY$(见表 10－8).

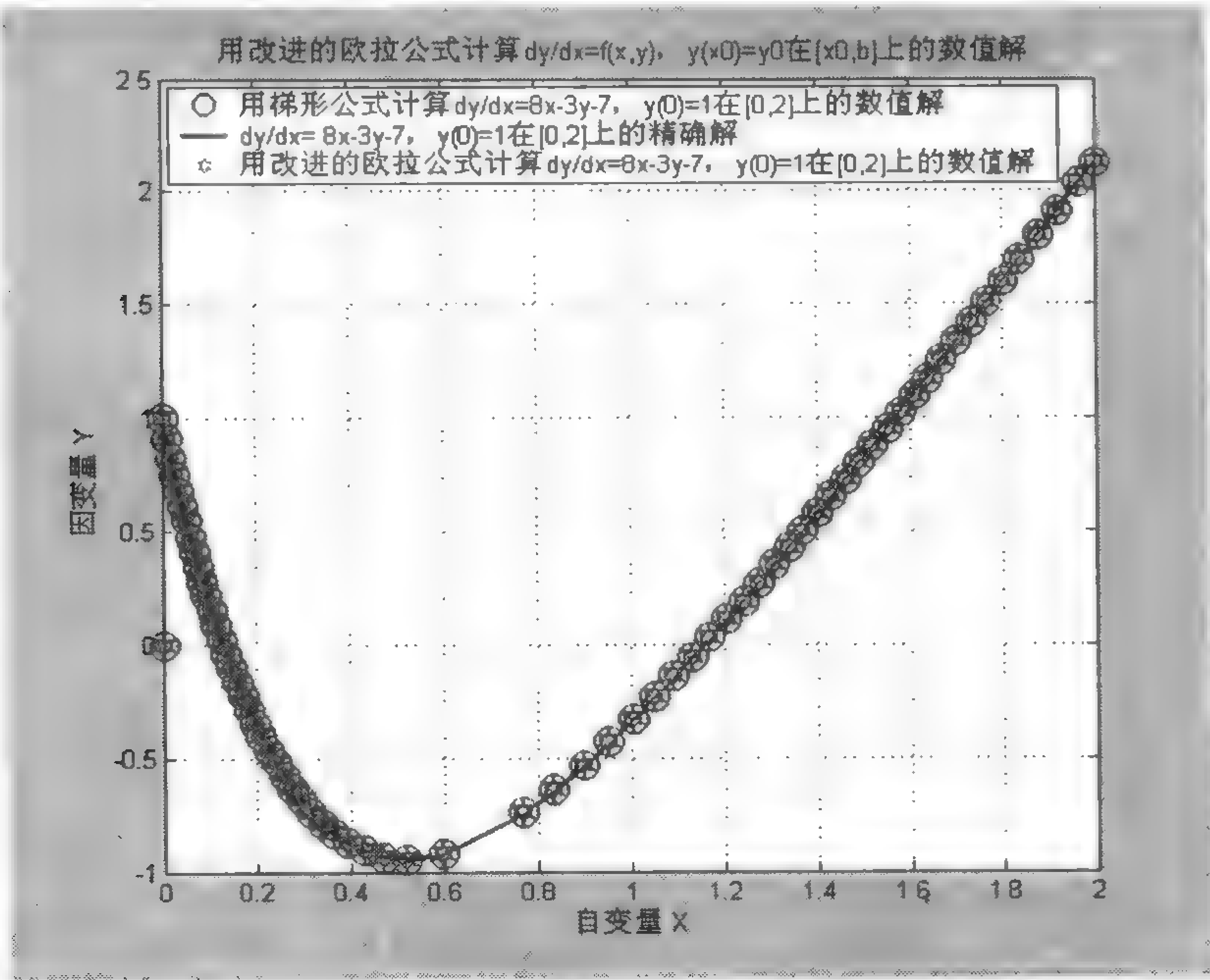


图 10－13 用梯形和改进的欧拉公式求解 $\frac{dy}{dx}=-3y+8x-7, y(0)=1$ 的图形

表 10－8 用梯形公式和改进欧拉公式求初值问题的数值解和精确解

n	H	X	Y	S_1	$juwY$	$xiwY$
1	0	0	1.000 0	1.000 0	0	0
2	0.009 1	0.009 1	0.910 2	0.910 2	−0.000 0	−0.000 0
3	0.008 5	0.017 6	0.829 7	0.829 7	−0.000 0	−0.000 0
4	0.008 2	0.025 8	0.754 7	0.754 6	−0.000 0	−0.000 0
5	0.008 0	0.033 8	0.683 0	0.683 0	−0.000 0	−0.000 1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
12	0.009 2	0.093 5	0.216 5	0.216 4	−0.000 1	−0.000 5
13	0.009 5	0.103 1	0.152 1	0.152 0	−0.000 1	−0.000 7
14	0.009 9	0.112 9	0.088 0	0.087 9	−0.000 1	−0.001 4
15	0.010 2	0.123 1	0.024 4	0.024 3	−0.000 1	−0.005 3
16	0.010 6	0.133 8	−0.038 7	−0.038 9	−0.000 1	0.003 7

续表

n	H	X	Y	S_1	$juwY$	$xiwY$
17	0.011 1	0.144 8	−0.101 4	−0.101 6	−0.000 2	0.001 5
18	0.011 5	0.156 3	−0.163 6	−0.163 8	−0.000 2	0.001 0
19	0.012 0	0.168 4	−0.225 3	−0.225 5	−0.000 2	0.000 8
20	0.012 6	0.181 0	−0.286 3	−0.286 5	−0.000 2	0.000 7
21	0.013 3	0.194 3	−0.346 6	−0.346 8	−0.000 2	0.000 6
⋮	⋮	⋮	⋮	⋮	⋮	⋮
42	0.042 5	1.091 8	−0.144 2	−0.151 2	−0.006 9	0.048 2
43	0.039 8	1.131 5	−0.056 9	−0.063 1	−0.006 2	0.109 2
44	0.037 6	1.169 2	0.027 7	0.022 1	−0.005 6	−0.201 8
45	0.035 9	1.205 1	0.110 0	0.104 9	−0.005 0	−0.045 8
46	0.034 6	1.239 7	0.190 6	0.1860	−0.004 6	−0.023 9
⋮	⋮	⋮	⋮	⋮	⋮	⋮
67	0.042 4	1.921 3	1.915 0	1.914 4	−0.000 6	−0.000 3
68	0.044 8	1.966 1	2.032 9	2.032 3	−0.000 6	−0.000 3
69	0.033 9	2.000 0	2.122 1	2.121 6	−0.000 5	−0.000 2

可以看出改进的欧拉公式和梯形公式的结果几乎相同,并且随着 x_n 的增加 y_n 累积误差的增长相当可观.

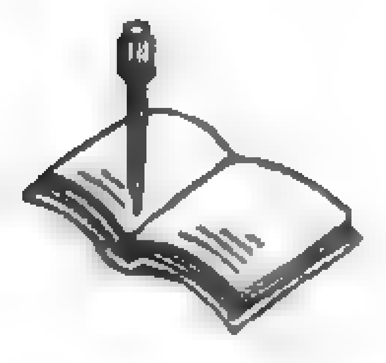
最后指出,上述**欧拉方法可以推广到解微分方程组**,如解

$$y'=f(x,y,z),z'=g(x,y,z),y(x_0)=y_0,z(x_0)=z_0 \tag{10.27}$$

的向前欧拉公式为

$$y_{n+1}=y_n+hf(x_n,y_n,z_n),z_{n+1}=z_n+hg(x_n,y_n,z_n),n=0,1,\cdots. \tag{10.28}$$

欧拉方法也可以**解高阶微分方程**,这只需先把它化为方程组.



习 题 10.4

1. 分别用梯形公式和向前欧拉公式求解区间[0,2]上的初值问题$\frac{dy}{dx}=-y,y(0)=1$,取精度为 10^{-1},并与精确解作比较,在同一个坐标系中画出图形.

2. 用梯形方法证明:初值问题$\frac{dy}{dx}=-y,y(0)=1$的数值解为$y_n=\left(\frac{2-h}{2+h}\right)^n$,并证明当$h\to0$时,它收敛于原初值问题的精确解$y=e^{-x}$.

3. 用向前欧拉公式和梯形公式求初值问题$y'=y-\frac{2x}{y},y(0)=1$的数值解($0\leqslant x\leqslant1$,步长$h=0.1$),并与精确解比较,在一个坐标系中作出它们的图形.

4. 分别用自适应梯形公式和向前欧拉公式求解区间[0,2]上的初值问题$\frac{dy}{dx}=-5y+8x-7,y(0)=1$,取精度为$10^{-1}$,并与精确解作比较,在同一个坐标系中作出图形.

5. 分别用改进欧拉公式和向前欧拉公式求解区间[0,2]上的初值问题$\frac{dy}{dx}=-y,y(0)=1$,取精度为10^{-1},并与精确解作比较,在同一个坐标系中作出图形.

6. 分别用改进欧拉公式、自适应梯形公式和梯形公式求初值问题$y'=y-\frac{2x}{y},y(0)=1$的数值解($0\leqslant x\leqslant1$,步长$h=0.1$),并与精确解比较,在一个坐标系中作出它们的图形.

10.5 龙格-库塔(Runge-Kutta)方法

10.5.1 龙格-库塔方法的基本思想

欧拉方法的基本思想是用差商代替导数.实际上,按照微分中值定理应有

$$\frac{y(x_{n+1})-y(x_n)}{h}=y'(x_n+\theta h),0<\theta<1. \tag{10.29}$$

注意到方程$y'=f(x,y)$就有

$$y(x_{n+1})=y(x_n)+hf(x_n+\theta h,y(x_n+\theta h)). \tag{10.30}$$

不妨记$\overline{K}=f(x_n+\theta h,y(x_n+\theta h))$,称为区间$[x_n,x_{n+1}]$上的平均斜率.可见给出一种平均斜率$\overline{K}$,(10.30)式就对应地导出一种算法.

向前欧拉公式简单地取$f(x_n,y_n)$为$\overline{K}$,精度自然很低.改进的欧拉公式可理解为$\overline{K}$取$f(x_n,y_n)$,$f(x_{n+1},\overline{y}_{n+1})$的平均值,其中$\overline{y}_{n+1}=y_n+hf(x_n,y_n)$,这种处理提高了精度.

如上分析启示我们,在区间$[x_n,x_{n+1}]$内预测$r(r>1)$个点处的斜率

$$\begin{cases}k_1=f(x_n,y_n),\\ k_2=f(x_n+a_2h,y_n+b_{21}hk_1),\\ \cdots\cdots\cdots\cdots\\ k_r=f\left(x_n+a_rh,y_n+h\sum_{j=1}^{r-1}b_{rj}k_j\right).\end{cases} \tag{10.31}$$

然后将它们加权平均作为平均斜率$\overline{K}$的近似值,即

$$y_{n+1}=y_n+h\sum_{i=1}^{r}c_ik_i, \tag{10.32}$$

就有可能构造出具有更高精度的求常微分方程初值问题(10.5)的计算公式.这就是**龙格-库塔方法的基本思想**.

10.5.2 二阶龙格-库塔方法及其 MATLAB 程序

(一) 二阶龙格-库塔公式

在区间$[x_n,x_{n+1}]$内取$r=2$个点,由(10.32)式和(10.31)式,得

$$\begin{cases}y_{n+1}=y_n+h(c_1k_1+c_2k_2),\\ k_1=f(x_n,y_n),\\ k_2=f(x_n+a_2h,y_n+b_{21}hk_1),0<a_2,b_{21}<1,\end{cases} \tag{10.33}$$

其中c_1,c_2,a_2,b_{21}为待定系数,看看如何确定它们使(10.33)式的精度尽量高.为此我们分析局部截断误差$y(x_{n+1})-y_{n+1}$,因为$y_n=y(x_n)$,所以(10.33)式可化为

$$\begin{cases}y_{n+1}=y(x_n)+h(c_1k_1+c_2k_2),\\ k_1=f(x_n,y(x_n))=y'(x_n),\\ k_2=f(x_n+a_2h,y(x_n)+b_{21}hk_1)\\ \quad=f(x_n,y(x_n))+a_2hf'_x(x_n,y(x_n))+\\ \quad\quad b_{21}hk_1f'_y(x_n,y(x_n))+O(h^2),0<a_2,b_{21}<1,\end{cases} \tag{10.34}$$

其中k_2在点$(x_n,y(x_n))$作了泰勒展开.(10.34)式又可表为

$$y_{n+1}=y(x_n)+h(c_1+c_2)y'(x_n)+a_2c_2h^2\left(f'_x+\frac{b_{21}}{a_2}ff'_y\right)+O(h^3). \tag{10.35}$$

因为在$y(x_{n+1})=y(x_n)+hy'(x_n)+\frac{h^2}{2}y''(x_n)+O(h^3)$中$y'=f,y''=f_x+ff_y$,可见为使误差$y(x_{n+1})-y_{n+1}=O(h^3)$,只需令

$$c_1+c_2=1,a_2c_2=1/2,b_{21}/a_2=1, \tag{10.36}$$

待定系数满足(10.36)的(10.33)式称**二阶龙格-库塔方法**.由于(10.36)式有4个未知数,3个方程,所以解不唯一.我们举出其中的三个为例:

取$c_1=c_2=\frac{1}{2},a_2=b_{21}=1$,则(10.33)式为改进的欧拉公式

$$\begin{cases}y_{n+1}=y_n+\frac{h}{2}(k_1+k_2),\\ k_1=f(x_n,y_n),\\ k_2=f(x_n+h,y_n+hk_1).\end{cases} \tag{10.37}$$

取 $c_1=0,c_2=1,a_2=b_{21}=\frac{1}{2}$,则(10.33)式为中点公式

$$\begin{cases} y_{n+1}=y_n+hk_2, \\ k_1=f(x_n,y_n), \\ k_2=f\left(x_n+\frac{1}{2}h,y_n+\frac{1}{2}hk_1\right). \end{cases} \tag{10.38}$$

取 $c_1=\frac{1}{4},c_2=\frac{3}{4},a_2=b_{21}=\frac{2}{3}$,则(10.33)式为休恩(Heun)公式

$$\begin{cases} y_{n+1}=y_n+\frac{h}{4}(k_1+3k_2), \\ k_1=f(x_n,y_n), \\ k_2=f\left(x_n+\frac{2}{3}h,y_n+\frac{2}{3}hk_1\right). \end{cases} \tag{10.39}$$

可以证明,在$[x_n,x_{n+1}]$内**只取2点的龙格-库塔公式精度最高为2阶.**

(二) 二阶龙格-库塔方法的MATLAB程序

根据二阶龙格-库塔方法的(10.33)式和(10.36)式编写的求解常微分方程初值问题(10.5)的MATLAB程序如下.

> **用二阶龙格-库塔方法求解常微分方程初值问题(10.5)的数值解的MATLAB主程序**
>
> **输入量**:*funfcn* 是函数 $f(x,y)$,*fun* 是初值问题的精确解函数,x_0和y_0是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,$\boldsymbol{C}=(c_1,c_2,a_2,b_{21})$ 是(10.36)式的参数组成的数组,h 是步长.
>
> **输出量**:向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值,向量 $\boldsymbol{Y}$ 的元素是利用(10.33)式和(10.36)式求出初值问题(10.5)在向量 $\boldsymbol{X}$ 的元素处的数值解,fxy 是在向量 $\boldsymbol{X}$ 的元素处的精确解,n 是自变量 x 取值的序号,wucha(k+1)=norm(Y(k+1)-Y(k)),wch(k)=norm(fxy(k)-Y(k)),并画出数值解和精确解的图形.

```
function [k,X,Y,fxy,wch,wucha,P]=RK2(funfcn,fun,x0,b,C,y0,h)
x=x0; y=y0;p=128; n=fix((b-x0)/h); fxy=zeros(p,1);
wucha=zeros(p,1); wch=zeros(p,1); X=zeros(p,1);
Y=zeros(p,length(y)); k=1; X(k)=x; Y(k,:)=y';
%绘图.
clc,x,h,y
    %计算
% fxy=fxy(:);
for k=2: n+1
```

```
        x = x + h;a2 = C(3);b21 = C(4);c1 = C(1);c2 = C(2);x1 = x + a2 * h;
        k1 = feval(funfcn,x,y);
        y1 = y + b21 * h * k1;k2 = feval(funfcn,x1,y1);
        fxy(k) = feval(fun,x);
        y = y + h * (c1 * k1 + c2 * k2);X(k) = x;Y(k,:) = y;k = k + 1;
        plot(X,Y,'mh',X,fxy,'bo')
        grid,xlabel('自变量 X'),ylabel('因变量 Y')
        legend('用二阶龙格 - 库塔方法计算 dy/dx = f(x,y),y(x0) = y0 在
[x0,b]上的数值解','y/dx = f(x,y),y(x0) = y0 的精确解 y = f(x)')
    end
    %计算误差.
    for k = 2: n + 1
    wucha(k) = norm(Y(k - 1) - Y(k)); wch(k) = norm(fxy(k) - Y(k));
    end
    X = X(1: k);Y = Y(1: k,:);fxy = fxy(1: k,:);
    n = 1: k;wucha = wucha(1: k,:);
    wch = wch(1: k,:); P = [n',X,Y,fxy,wch,wucha];
```

例 10.5.1 用二阶龙格 - 库塔方法求初值问题

$$\begin{cases}\dfrac{dy}{dx}=1-\dfrac{2xy}{1+x^2},0\leqslant x\leqslant 2,\\ y|_{x=0}=0\end{cases}$$

的数值解,取 $c_1=\dfrac{1}{4},c_2=\dfrac{3}{4},a_2=b_{21}=\dfrac{2}{3},h=\dfrac{1}{4}$,并计算与精确解的误差,画出精确解和数值解的图形.

解 (1) 先求精确解. 输入程序

```
>> y = dsolve('(Dy) + (2 * x * y)/(1 + x^2) - 1 = 0','y(0) = 0', 'x')
```

运行后屏幕显示常微分方程在给定初始条件下的精确解 y 如下

```
y =
  (x + 1/3 * x^3)/(1 + x^2)
```

(2) 用二阶龙格 - 库塔方法求初值问题的数值解,并计算误差,画出精确解和数值解的图形.

① 对此问题的二阶龙格 - 库塔公式(10.33)的具体形式为

$$\begin{cases}x=x_0+jh,j=0,1,2,\cdots,\left[\dfrac{b-x_0}{h}\right],\\ y_{n+1}=y_n+h(c_1k_1+c_2k_2),n=0,1,2,\cdots,\\ k_1=1-\dfrac{2x_ny_n}{1+x_n^2},\\ k_2=1-\dfrac{2(x_n+a_2h)(y_n+b_{21}hk_1)}{1+(x_n+a_2h)^2},0<a_2,b_{21}<1.\end{cases}$$

② 编写并保存名为 funfcn.m 和 fun.m 的 M 文件如下

```
function f = funfcn(x,y)
f =1 -(2 * x * y)/(1 +x^2);
function y = fun (x)
y =(x +1/3 * x^3)/(1 +x^2);
```

③ 在 MATLAB 工作窗口输入下面的程序

```
>> x0 =0;b =2;C =[1/4,3/4,2/3,2/3];y0 =0;h =1/4;
[k,X,Y,fxy,wch,wucha,P] = RK2(@ funfcn,@ fun,x0,b,C,y0,h)
```

④ 将运行后计算的结果列入表 10-9,画出精确解和数值解的图形(见图 10-14).

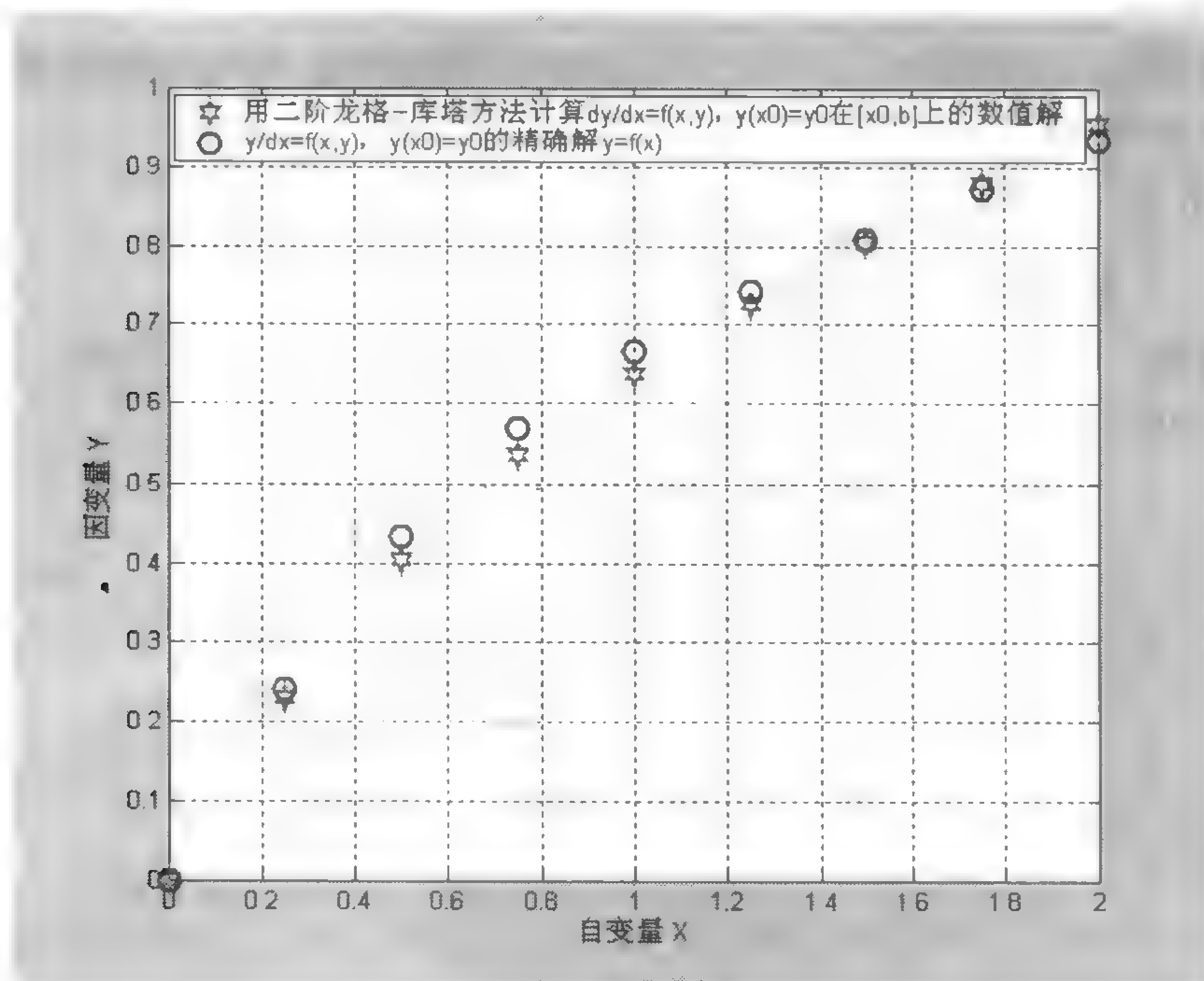


图 10-14 用二阶龙格-库塔方法求初值问题

表 10-9 用二阶龙格-库塔方法求初值问题

n	x_n	y_n	$y(x_n)$	$\\| y(x_n) - y_n \\|_2$	$\\| y_{n+1} - y_n \\|_2$
1	0	0	0	0	0
2	0.250 0	0.227 8	0.240 2	0.012 4	0.227 8
3	0.500 0	0.403 4	0.433 3	0.029 9	0.175 6

续表

n	x_n	y_n	$y(x_n)$	$\|y(x_n)-y_n\|_2$	$\|y_{n+1}-y_n\|_2$
4	0.750 0	0.534 8	0.570 0	0.035 2	0.131 4
5	1.000 0	0.637 9	0.666 7	0.028 8	0.103 1
6	1.250 0	0.725 2	0.741 9	0.016 7	0.087 3
7	1.500 0	0.804 2	0.807 7	0.003 4	0.079 1
8	1.750 0	0.879 4	0.870 5	0.008 9	0.075 1
9	2.000 0	0.952 9	0.933 3	0.019 6	0.073 5

从图 10－14 和表 10－9 可见，用二阶龙格－库塔方法求初值问题的数值解明显地改善了精度.

10.5.3 三阶龙格－库塔方法及其 MATLAB 程序

(一) 三阶龙格－库塔方法

要进一步提高精度，必须取更多的点，如果在区间 $[x_n, x_{n+1}]$ 内取 $r=3$ 个点，由(10.32)式和(10.31)式，得**三阶龙格－库塔方法的形式**为

$$\begin{cases} y_{n+1}=y_n+h(c_1k_1+c_2k_2+c_3k_3), \\ k_1=f(x_n, y_n), \\ k_2=f(x_n+a_2h, y_n+b_{21}hk_1), \\ k_3=f[x_n+a_3h, y_n+h(b_{31}k_1+b_{32}k_2)], \end{cases} \tag{10.40}$$

其中

$$\begin{cases} c_1+c_2+c_3=1, \\ a_2c_2+a_3c_3=\dfrac{1}{2}, \\ a_2^2c_2+a_3^2c_3=\dfrac{1}{3}, \\ a_2b_{32}c_3=\dfrac{1}{6}, \\ a_2=b_{21}, \\ a_3=b_{31}+b_{32}. \end{cases} \tag{10.41}$$

因为方程组(10.41)含八个未知数和六个方程，所以有多组不同的解，因此有多种不同的三阶龙格－库塔公式. 取 $c_1=\dfrac{1}{6}, c_2=\dfrac{2}{3}, c_3=\dfrac{1}{6}, a_2=b_{21}=\dfrac{1}{2}, a_3=1, b_{31}=-1, b_{32}=2$，则(10.40)式为**常用的三阶龙格－库塔公式**.

(二) 三阶龙格－库塔方法的 MATLAB 程序

根据三阶龙格－库塔方法(10.40)式和(10.11)式编写的求解常微分方程初值问题(10.5)的 MATLAB 程序如下.

用三阶龙格－库塔方法求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序

输入量：*funfcn* 是函数 $f(x,y)$，*fun* 是初值问题的精确解函数，x_0 和 y_0 是初值 $y(x_0)=y_0$，b 是自变量 x 的最大值，$\boldsymbol{C}=(c_1,c_2,c_3,a_2,a_3,b_{21},b_{31},b_{32})$ 是(10.40)式的参数组成的数组，h 是步长.

输出量：向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值，向量 $\boldsymbol{Y}$ 的元素是利用(10.40)式和(10.41)式求出初值问题(10.5)在向量 $\boldsymbol{X}$ 的元素处的数值解，*fxy* 是在向量 $\boldsymbol{X}$ 的元素处的精确解，n 是自变量 x 取值的序号，wucha(k+1) = norm(Y(k+1) - Y(k))，wch(k) = norm(fxy(k) - Y(k))，并画出数值解和精确解的图形.

```
function [k,X,Y,fxy,wch,wucha,P] = RK3(funfcn,fun,x0,b,C,y0,
h)
x = x0; y = y0;p = 128; n = fix((b - x0)/h);
fxy = zeros(p,1);wucha = zeros(p,1);
wch = zeros(p,1); X = zeros(p,1);
Y = zeros(p,length(y)); k = 1; X(k) = x; Y(k,:) = y';
% 绘图.
clc,x,h,y
    % 计算
% fxy = fxy(:);
for k = 2: n + 1
x = x + h;a2 = C(4); a3 = C(5);b21 = C(6); b31 = C(7);
b32 = C(8);c1 = C(1); c2 = C(2); c3 = C(3); x1 = x + a2 * h;
x2 = x + a3 * h; k1 = feval(funfcn,x,y); y1 = y + b21 * h * k1;
k2 = feval(funfcn,x1,y1);
y2 = y + b31 * h * k1 + b32 * h * k2; k3 = feval(funfcn,x2,y2);
fxy(k) = feval(fun,x);
y = y + h * (c1 * k1 + c2 * k2 + c3 * k3); X(k) = x; Y(k,:) = y; k = k + 1;
plot(X,Y,'rp',X,fxy,'bo'),
grid,xlabel('自变量 X'), ylabel('因变量 Y')
legend('用三阶龙格－库塔方法计算 dy/dx = f(x,y),y(x0) = y0 在[x0,
b]上的数值解','y/dx = f(x,y),y(x0) = y0 的精确解 y = f(x)')
end
% 计算误差.
```

```
for k =2: n +1
wucha(k) =norm(Y(k -1) -Y(k)); wch(k) =norm(fxy(k) -Y(k));
end
X =X(1: k); Y =Y(1: k,:);fxy =fxy(1: k,:);n =1: k;
wucha =wucha(1: k,:);wch =wch(1: k,:); P =[n ',X,Y,fxy,wch,
wucha];
```

例 10.5.2 用常用的三阶龙格 - 库塔公式求初值问题

$$\begin{cases}\dfrac{dy}{dx}=1-\dfrac{2xy}{1+x^2},\ 0\leqslant x\leqslant 2,\\ y\Big|_{x=0}=0\end{cases}$$

的数值解,取 $c_1=\dfrac{1}{6},c_2=\dfrac{2}{3},c_3=\dfrac{1}{6},a_2=b_{21}=\dfrac{1}{2},a_3=1,b_{31}=-1,b_{32}=2,h=\dfrac{1}{4}$,并计算与精确解的误差,画出精确解和数值解的图形.

解 (1) 对此问题的常用的三阶龙格 - 库塔公式的具体形式为

$$\begin{cases}x=x_0+jh,j=0,1,2,\cdots,\left[\dfrac{b-x_0}{h}\right],\\ y_{n+1}=y_n+h(c_1k_1+c_2k_2+c_3k_3),n=0,1,2,\cdots,\\ k_1=1-\dfrac{2x_ny_n}{1+x_n^2},\\ k_2=1-\dfrac{2(x_n+a_2h)(y_n+b_{21}hk_1)}{1+(x_n+a_2h)^2},\\ k_3=1-\dfrac{2(x_n+a_3h)(y_n+b_{31}hk_1+b_{32}hk_2)}{1+(x_n+a_3h)^2}.\end{cases}$$

取 $c_1=\dfrac{1}{6},c_2=\dfrac{2}{3},c_3=\dfrac{1}{6},a_2=b_{21}=\dfrac{1}{2},a_3=1,b_{31}=-1,b_{32}=2.$

(2) 编写并保存名为 funfcn.m 和 fun.m 的 M 文件如下

```
function f =funfcn(x,y)
f =1 -(2 *x *y)/(1 +x^2);
function y =fun (x)
y =(x +1/3 *x^3)/(1 +x^2);
```

(3) 在 MATLAB 工作窗口输入下面的程序

```
>> x0 =0;b =2; c1 =1/6;c2 =2/3;c3 =1/6;a2 =1/2;
a3 =1;b21 =1/2;b31 = -1;b32 =2;
C =[c1,c2,c3,a2, a3,b21,b31,b32];y0 =0;h =1/4;
[k,X,Y,fxy,wch,wucha,P] =RK3(@funfcn,@fun,x0,b,C,y0,h)
```

将运行后计算的结果如表 10 - 10,画出精确解和数值解的图形见图 10 - 15.

表 10 - 10 用常用的三阶龙格 - 库塔公式求初值问题

n	x_n	y_n	$y(x_n)$	$\Vert y(x_n)-y_n\Vert_2$	$\Vert y_{n+1}-y_n\Vert_2$
1	0	0	0	0	
2	0.250 0	0.229 3	0.240 2	0.010 9	0.229 3
3	0.500 0	0.407 0	0.433 3	0.026 3	0.177 7
4	0.750 0	0.539 4	0.570 0	0.030 6	0.132 4
5	1.000 0	0.642 6	0.666 7	0.024 0	0.103 2
6	1.250 0	0.729 5	0.741 9	0.012 4	0.086 8
7	1.500 0	0.807 9	0.807 7	0.000 2	0.078 5
8	1.750 0	0.882 5	0.870 5	0.012 0	0.074 6
9	2.000 0	0.955 5	0.933 3	0.022 2	0.073 0

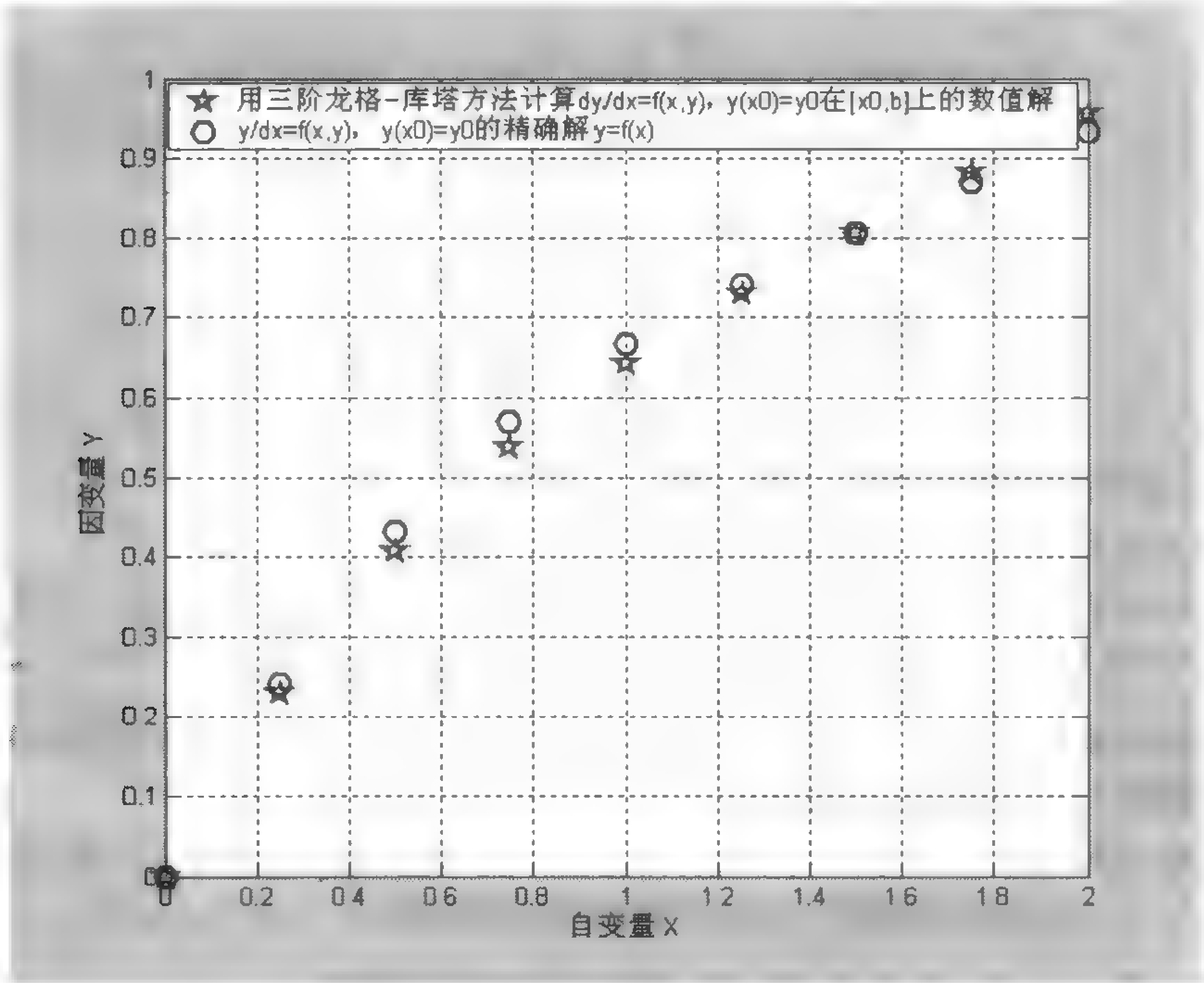


图 10 - 15 常用的三阶龙格 - 库塔公式求初值问题

从图 10 - 15 和表 10 - 10 可见,用三阶龙格 - 库塔方法比二阶龙格 - 库塔方法求初值问题的数值解的精度更高.

10.5.4 四阶龙格－库塔方法及其MATLAB程序

(一)四阶龙格－库塔方法

如果在区间$[x_n,x_{n+1}]$内取$r=4$个点,由(10.32)式和(10.31)式,得**四阶龙格－库塔方法的格式**为

$$\begin{cases}y_{n+1}=y_n+h(c_1k_1+c_2k_2+c_3k_3+c_4k_4),\\k_1=f(x_n,y_n),\\k_2=f(x_n+a_2h,y_n+b_{21}hk_1),\\k_3=f[x_n+a_3h,y_n+h(b_{31}k_1+b_{32}k_2)],\\k_4=f[x_n+a_4h,y_n+h(b_{41}k_1+b_{42}k_2+b_{43}k_3)],\end{cases}\tag{10.42}$$

其中待定系数c_i,a_i,b_{ij}共13个,经过与推导二阶龙格－库塔公式类似、但更复杂的计算,得到使局部误差$y(x_{n+1})-y_{n+1}=O(h^5)$的11个方程.取既满足这些方程、又较简单的一组$c_i,a_i,b_{ij}$,可得

$$\begin{cases}y_{n+1}=y_n+\dfrac{h}{6}(k_1+2k_2+2k_3+k_4),\\k_1=f(x_n,y_n),\\k_2=f\left(x_n+\dfrac{h}{2},y_n+h\dfrac{k_1}{2}\right),\\k_3=f\left(x_n+\dfrac{h}{2},y_n+h\dfrac{k_2}{2}\right),\\k_4=f(x_n+h,y_n+hk_3).\end{cases}\tag{10.43}$$

这就是**常用的四阶龙格－库塔公式**.

(二)四阶龙格－库塔方法的MATLAB程序

分别根据一般四阶龙格－库塔方法(10.42)和常用四阶龙格－库塔方法(10.43)编写的求常微分方程初值问题(10.5)的两种MATLAB程序如下.

用一般四阶龙格－库塔方法(10.42)式求解常微分方程初值问题(10.5)的数值解的MATLAB主程序1

输入量:*funfcn*是函数$f(x,y)$,*fun*是初值问题的精确解函数,x_0和y_0是初值$y(x_0)=y_0$,b是自变量x的最大值,h是步长,$\boldsymbol{C}=(\mathrm{c}_1,\mathrm{c}_2,\mathrm{c}_3,\mathrm{c}_4,\mathrm{a}_2,\mathrm{a}_3,\mathrm{a}_4,\mathrm{b}_{21},\mathrm{b}_{31},\mathrm{b}_{32},\mathrm{b}_{41},\mathrm{b}_{42},\mathrm{b}_{43})$是(10.42)式的参数组成的数组.

输出量:向量$\boldsymbol{X}$的元素是自变量x的取值,向量$\boldsymbol{Y}$的元素是利用(10.42)式求出初值问题(10.5)在向量$\boldsymbol{X}$的元素处的数值解,*fxy*是在向量$\boldsymbol{X}$的元素处的精确解,n是自变量x取值的序号,wucha(k+1)=norm(Y(k+1)-Y(k)),wch(k)=norm(fxy(k)-Y(k)),并画出数值解和精确解的图形.

```
function [k,X,Y,fxy,wch,wucha,P] = RK4 (funfcn,fun,x0,b,C,y0,
h)
x = x0; y - y0;p = 128; n = fix((b - x0)/h); fxy = zeros(p,1);
wucha = zeros(p,1); wch = zeros(p,1); X = zeros(p,1);
Y = zeros(p,length(y)); k = 1; X(k) = x; Y(k,:) = y';
%绘图.
clc,x,h,y
    %计算
%fxy = fxy(:);
for k - 2: n + 1
x = x + h;a2 = C(5); a3 = C(6); a4 = C(7);b21 = C(8); b31 = C(9);
b32 = C(10); b41 = C(11); b42 = C(12); b43 = C(13);c1 = C(1);
c2 - C(2); c3 = C(3); c4 = C(4); x1 = x + a2 * h; x2 = x + a3 * h;
x3 = x + a4 * h; k1 = feval(funfcn,x,y); y1 = y + b21 * h * k1;
k2 = feval(funfcn,x1,y1); y2 = y + b31 * h * k1 + b32 * h * k2;
k3 = feval(funfcn,x2,y2);
y3 = y + b41 * h * k1 + b42 * h * k2 + b43 * h * k3;
k4 = feval(funfcn,x3,y3); fxy(k) = feval(fun,x);
y = y + h * (c1 * k1 + c2 * k2 + c3 * k3 + c4 * k4);
X(k) = x; Y(k,:) = y; k = k + 1; plot(X,Y,'rp',X,fxy,'bo'), grid
xlabel('自变量 X'), ylabel('因变量 Y')
legend('用四阶龙格 - 库塔方法计算 dy/dx = f(x,y),y(x0) = y0 在[x0,
b]上的解','y/dx = f(x,y),y(x0) = y0 的精确解 y = f(x)')
end
%计算误差.
for k = 2: n + 1
wucha(k) = norm(Y(k - 1) - Y(k)); wch(k) = norm(fxy(k) - Y(k));
end
X = X(1: k); Y = Y(1: k,:);fxy = fxy(1: k,:);
n = 1: k;wucha = wucha(1: k,:);
wch = wch(1: k,:);,P = [n',X,Y,fxy,wch,wucha];
```

另外，还可以根据常用四阶龙格－库塔公式(10.43)编写求解常微分方程初值问题(10.5)的 MATLAB 程序如下：

用常用四阶龙格－库塔方法求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序 2

输入量：*funfcn* 是函数 $f(x,y)$，x_0和 y_0是初值 $y(x_0)=y_0$，b 是自变量 x 的最大值，h 是步长，

输出量：x 是自变量 x 的取值，y 是利用(10.43)式求出初值问题(10.5)在 x 处的数值解.

```
function [x,y] = RKc4(x0,b,y0,h)
for x = x0:h:b
        k1 = funfcn(x0,y0);
        k2 = funfcn(x0 + h/2,y0 + h * k1/2);
        k3 - funfcn(x0 + h/2,y0 + h * k2/2);
        k4 = funfcn(x0 + h,y0 + h * k3);
        y - y0 + h * (k1 + 2 * k2 + 2 * k3 + k4)/6;
        x = x0 + h;[x,y];x0 = x;y0 = y;
end
[x,y];
```

例 10.5.3 用常用的四阶龙格－库塔公式求初值问题

$$\begin{cases}\dfrac{dy}{dx}=1-\dfrac{2xy}{1+x^2},0\leqslant x\leqslant 2,\\ y\Big|_{x=0}=0\end{cases}$$

的数值解，$h=\dfrac{1}{4}$，并计算与精确解的误差，在同一图形窗口画出例 10.5.1、例 10.5.2 和本例精确解和数值解的图形.

解 方法 1 (1) 对此问题的常用的四阶龙格－库塔公式(10.43)的具体形式为

$$\begin{cases}x=x_0+jh,j=0,1,2,\cdots,\left[\dfrac{b-x_0}{h}\right],\\ y_{n+1}=y_n+h(c_1k_1+c_2k_2+c_3k_3+c_4k_4),n=0,1,2,\cdots,\\ k_1=1-\dfrac{2x_ny_n}{1+x_n^2},\\ k_2=1-\dfrac{2(x_n+a_2h)(y_n+b_{21}hk_1)}{1+(x_n+a_2h)^2},\\ k_3=1-\dfrac{2(x_n+a_3h)(y_n+b_{31}hk_1+b_{32}hk_2)}{1+(x_n+a_3h)^2},\\ k_4=1-\dfrac{2(x_n+a_4h)(y_n+b_{41}hk_1+b_{42}hk_2+b_{43}hk_3)}{1+(x_n+a_4h)^2}.\end{cases}$$

取 $b_{31}=b_{41}=b_{42}=0,b_{43}=1,c_1=\dfrac{1}{6},c_2=\dfrac{1}{3},c_3=\dfrac{1}{3},c_4=\dfrac{1}{6},a_4=1,a_2=a_3=b_{21}=b_{32}=\dfrac{1}{2}$.

(2) 编写并保存名为 funfcn.m 和 fun.m 的 MATLAB 程序.

(3) 在 MATLAB 工作窗口输入下面的程序

```
>> x0 =0;b=2; y0 =0;h=1/4;
subplot(3,1,1)
C=[1/4,3/4,2/3,2/3];
[k,X,Y,fxy,wch,wucha,P]=RK2(@funfcn,@fun,x0,b,C,y0,h)
subplot(3,1,2)
c1 =1/6;c2 =2/3;c3 =1/6;a2 =1/2; a3 =1;
b21 =1/2;b31 = -1;b32 =2;
C=[c1,c2,c3,a2, a3,b21,b31,b32];
[k,X,Y,fxy,wch,wucha,P]=RK3(@funfcn,@fun,x0,b,C,y0,h)
subplot(3,1,3)
c1 =1/6;c2 =1/3;c3 =1/3;c4 =1/6;a2 =1/2; a3 =1/2;
a4 =1;b21 =1/2;b31 =0;b32 =1/2; b41 =0;b42 =0;b43 =1;
C=[c1,c2,c3, c4,a2, a3, a4,b21,b31,b32,b41,b42,b43];
[k,X,Y,fxy,wch,wucha,P]=RK4(@funfcn,@fun,x0,b,C,y0,h)
```

将运行后计算的结果如表 10 - 11,画出精确解和数值解的图形见图 10 - 16.

表 10 - 11 用常用的四阶龙格 - 库塔公式求初值问题

n	x_n	y_n	$y(x_n)$	$\|y(x_n)-y_n\|_2$	$\|y_{n+1}-y_n\|_2$
1	0	0	0	0	0
2	0.250 0	0.229 2	0.240 2	0.011 0	0.229 2
3	0.500 0	0.406 7	0.433 3	0.026 7	0.177 5
4	0.750 0	0.539 0	0.570 0	0.031 0	0.132 4
5	1.000 0	0.642 3	0.666 7	0.024 4	0.103 2
6	1.250 0	0.729 2	0.741 9	0.012 7	0.086 9
7	1.500 0	0.807 7	0.807 7	0.000 0	0.078 5
8	1.750 0	0.882 3	0.870 5	0.011 8	0.074 6
9	2.000 0	0.955 3	0.933 3	0.022 0	0.073 0

方法 2 在 MATLAB 工作窗口输入下面的程序

```
>> x0 =0;b=2; y0 =0;h=1/4;
[x,y]=RKc4(x0,b,y0,h)
```

运行后输出结果(略).

需要说明的是,当 $r=1,2,3,4$ 时,龙格 - 库塔公式(10.31)式和(10.32)式的最高阶数恰好是 r;当 $r>4$ 时,龙格 - 库塔公式的最高阶数不是 r,如 $r=5$ 时,龙格 - 库塔公式的最高阶数仍是 4,当 $r=6$ 时,龙格 - 库塔公式的最高阶数是 5. 因为龙格 - 库塔方法的导出基于泰勒展开,所以它要求所求问题的解具有较高的光滑度. 当解充分光滑时,四阶龙格 - 库塔方法确实优于二阶或三阶龙格 -

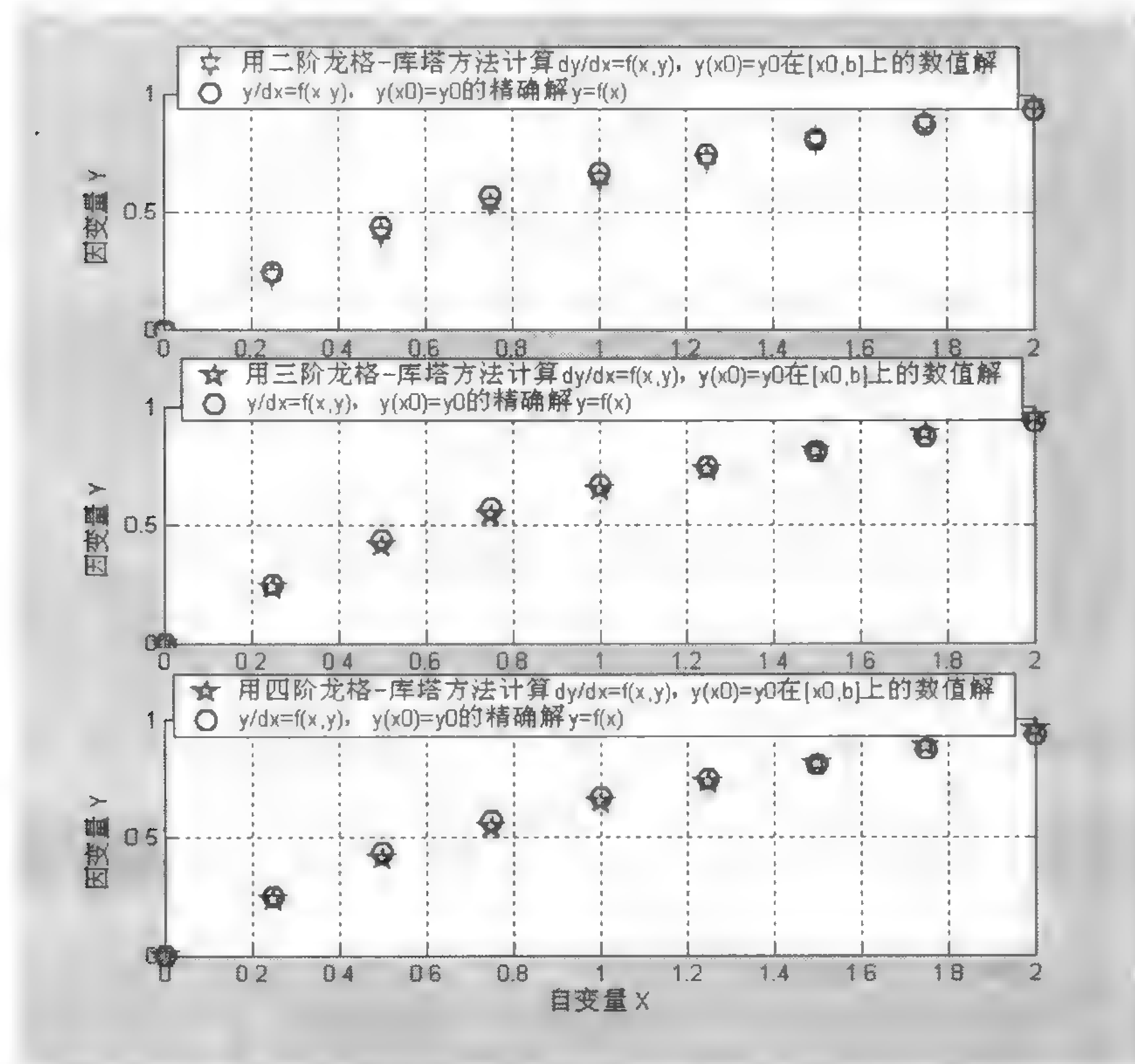


图 10-16 用常用的四阶龙格-库塔公式求初值问题

库塔方法. 对于大量的实际问题，四阶龙格-库塔方法一般可以达到精度要求. 如果解的光滑性差，则用四阶龙格-库塔方法求初值问题(10.5)的数值解的效果可能不如二阶或三阶龙格-库塔方法. 因此，实际计算时，需要根据问题的具体情况来选择合适的算法.

10.5.5 自适应龙格-库塔方法及其 MATLAB 程序

(一) 自适应龙格-库塔方法

一般来说，步长越小，截断误差就越小. 但是，随着步长的缩小，由起始点计算到给定点所需的步数就增加了. 步数的增加，不但引起计算量的增大，而且可能导致舍入误差的严重积累. 因此，在实际计算时，需要选择适当的步长，在满足精度要求的前提下，尽可能地减少计算步数. 一种有效的措施就是在计算过程中，根据初值问题(10.5)的解函数 $y(x)$ 的变化的不均匀性，在计算的过程中自动地调整步长，即**自适应算法**. 自适应算法是提高计算效率的有效算法. 因为前面介绍过的解微分方程的数值解法需要使用等距节点，自适应算法是在解函数

$y(x)$性态好(如函数值变化不大等)的子区间上步长不变,而在另一些性态不好(如函数值变化巨大等)的子区间上缩小步长,插入较多的节点. **自适应算法的基本思想**是通过数值方法辨识解函数 $y(x)$ 的性态,进而进行合理地剖分. 常用的自适应算法之一是自适应龙格－库塔方法,其具体方法如下.

设用 r 阶龙格－库塔方法计算 y_{n+1}. 我们首先从 y_n出发,以步长 h 计算一步所得 $y(x_{n+1})$的近似值为 $y_{n+1}^{(h)}$,以步长$\frac{h}{2}$计算两步得 $y(x_{n+1})$的近似值为 $y_{n+1}^{(\frac{h}{2})}$. 由于 r 阶龙格－库塔公式的局部截断误差为 $O(h^{r+1})$,且 $y^{(r+1)}(x)$在小区间$[x_n, x_{n+1}]$上变化不大,故有

$$y(x_{n+1})-y_{n+1}^{(h)}\approx ch^{r+1}, \tag{10.44}$$

$$y(x_{n+1})-y_{n+1}^{(\frac{h}{2})}\approx 2c\left(\frac{h}{2}\right)^{r+1}. \tag{10.45}$$

将(10.45)乘以 2^r减(10.44),得

$$(2^r-1)y(x_{n+1})-2^r y_{n+1}^{(\frac{h}{2})}+y_{n+1}^{(h)}\approx 0.$$

从而有

$$|y(x_{n+1})-y_{n+1}^{(h)}|\approx\frac{1}{2^r-1}|y_{n+1}^{(h)}-y_{n+1}^{(\frac{h}{2})}|=\Delta. \tag{10.46}$$

这样,可以根据事后误差估计(10.46)式中 Δ 的大小来选择合适的步长.

自适应龙格－库塔方法按以下过程进行:先以 h 为步长,从点 x_n出发计算一步得 $y(x_n+h)$的近似值为 $y_{n+1}^{(h)}$,再以$\frac{h}{2}$为步长,从点 x_n 出发计算两步得 $y(x_n+h)$的又一个近似值为 $y_{n+1}^{(\frac{h}{2})}$. 对于预先给定的容限 ε,分下列三种情况处理:

(1) 如果 $\Delta<\varepsilon$,且相差不多,则表明步长是合适的,取 $y(x_{n+1})\approx y_{n+1}=y_{n+1}^{(h)}$,并以 h 为步长计算点 x_{n+2}处的近似值 y_{n+2};

(2) 如果 $\Delta\ll\varepsilon$,则表明步长过小,应该反复将步长加倍进行计算,直到 $\Delta>\varepsilon$ 为止. 此时前一次的计算结果为 y_{n+1},步长即为是合适的步长,以此步长继续计算;

(3) 如果 $\Delta>\varepsilon$,则表明步长过大,应该将步长折半再进行计算,直到满足 $\Delta<\varepsilon$ 为止.

(二) 常微分方程(组)的刚性和非刚性

根据常微分方程(组)的描述问题的性质,可将常微分方程划分为刚性方程和非刚性方程. 这里对刚性方程只给直观的描述,更详细的内容请参见有关书籍(如唐立山等编著的《非数值并行算法(第一册)》,由科学出版社 1994 年出版). 自然界各种现象发生的每一过程都是及其复杂的,往往包含许多子过程及其它们之间的相互作用,其中有些子过程表现为快的变化,而有些则相对慢些,且变

化的速度可以相差非常大的数量级,相应地描述这些子过程的常微分方程(组)的解中也将包含快变化和慢变化的分量,如果在一个过程中的快变量过程与慢变量过程的变化速度相差非常大,在数学上称这种过程具有"刚性",而描述这种过程的常微分方程(组)称为**刚性方程(组)**,也称为**病态方程(组)**或**坏条件方程(组)**. 否则称为**非刚性方程(组)**. 对于一阶常系数线性微分方程组 $\dot{\boldsymbol{y}}=\boldsymbol{A}\boldsymbol{y}+\boldsymbol{f}$,如果它的雅可比矩阵 $\boldsymbol{A}_{n\times n}$ 的特征值相差十分悬殊,则称此微分方程组是**刚性的**. "刚性"问题在控制系统工程、电子网络、生物学,物理学及化学动力学过程中经常遇到. 正是由于这种性质,使得传统的常微分方程(组)数值解法遇到极大的困难. 在 MATLAB 系统中,提供了数值解刚性和非刚性常微分方程的程序. 根据微分方程组刚性的不同情况(特征值相差悬殊的情况),应该采用不同的求解函数,才能得到较好的效果.

(三) 解常微分方程初值问题的 MATLAB 库函数

在 MATLAB 系统中,有七个专门用于解常微分方程初值问题的库函数,具体的库函数名、解决问题的类型、精度及其适用范围参见表 10 - 12.

表 10 - 12 解常微分方程初值问题的库函数

odeij	问题类型	精度	适用对象
ode45	非刚性	中等	多数情况下可优先适用,但不能用来解刚性问题
ode113	非刚性	低到高	不能解刚性问题,当误差容限要求严格时,效果较 ode45 好
ode23	非刚性	较低	可用来解中等刚性问题,或误差允许范围较宽的问题
ode15s	刚性	低到高	可用来解刚性问题,当采用 ode45 失败或效果很差时,可考虑适用
ode23s	刚性	低	可用来解刚性问题,当误差容限较宽时,效果较 ode15s 好
ode23tb	刚性	低	可用来解刚性问题,当误差容限较宽时,效果较 ode15s 好
ode23t	适度刚性	低	可用来解刚性问题,但要求无数值衰减

表 10 - 12 中 ode23,ode45,ode23s 等库函数主要采用自适应龙格-库塔方法. 其中 ode23 系列为采用二阶、三阶龙格-库塔方法,ode45 系列为采用四阶、五阶龙格-库塔方法. 一般来说,ode45 比 ode23 的积分段少,运算速度更快一些. 表 10 - 12 中每个库函数的详细说明,请在 MATLAB 工作窗口输入命令:help 函数名进行查询.

表 10 - 12 中每个库函数的一般调用格式有如下几种:

调用格式一:[T,Y] = odeij (@ODEFUN,TSPAN,Y0)

输入量:(1) ODEFUN 是由待解的微分方程(组)$y'=f(t,y)$中函数$f(t,y)$写成 M 函数文件名;Y_0是初始条件;用于解 n 个未知函数的方程组时,Y_0和 Y 均为 n 维向量,M 文件中的待解方程组应以 Y 的分量形式写成.

(2) TSPAN = [T0 TFINAL]是积分区间,其中 T_0,T_{FINAL}分别为自变量 t 的初值和终值;一般情况下输入 TSPAN = [T0 TFINAL];如果要求输出在指定点 $t=T_0,T_1,\cdots,T_{FINAL}$处的微分方程的数值解 $Y=Y_0,Y_1,\cdots,Y_{FINAL}$,则输入 TSPAN = [T0 T1 ... TFINAL];等步长时用 t = T0:k:TFINAL,输出 Y 是在$[T_0,T_{FINAL}]$上的等分点 $t=T_0+nk(n=0,1,2,\cdots,m)$处的数值解.

(3) t 的步长是程序根据误差限自动选定的.

(4) odeij 表示表 10－12 中每个库函数名.

输出量:T 和 Y 是自变量 t 和 $y'=f(t,y)$在自变量 t 的值(列向量)T 处的数值解(列向量).

调用格式二:[T,Y] = odeij (@ODEFUN,TSPAN,Y0,OPTIONS)

其中(1) 输入量 ODEFUN,TSPAN,Y0,odeij 和输出量 Y 和 T 同上;

(2) OPTIONS 是用来设置一些可选的参数值,缺省时相当调用格式一.一般地被用于设定误差限,即数量的相对误差限'RelTol'(缺省值为 1e－3)和向量的各分量的绝对误差限'AbsTol'(各分量的缺省值为 1e－6).OPTIONS 是用函数 ODESET 创建的,详细内容请察看 help ODESET.

调用格式三:[T,Y] = odeij (@ODEFUN,TSPAN,Y0,OPTIONS,P1,P2,…)

其中(1) 输入量 ODEFUN,TSPAN,Y0,odeij,OPTIONS 和输出量 Y 和 T 同上;

(2) P1,P2,...的作用是将附加参数 $P_1,P_2,...$ 传递给 ODE 文件(如 ODEFUN(T,Y,P1,P2 ...))和在 OPTIONS 中的所有函数.如果没有 OPTIONS 参数设置,则用 OPTIONS = [ ]代替,以便正确传递参数.如果 TSPAN 或 Y_0没有被指定,则 ODEij 函数调用 ODE 文件,令[TSPAN,Y0, OPTIONS] = ODEFUN([ ],[ ],'init')来获得在 ODEij 函数调用时没有提供的相关参数值,甚至调用参数列表中末尾的没有的参数,如格式 odeij (@ODEFUN).

调用格式四:[T,Y,TE,YE,IE] = odeij(@ODEFUN,TSPAN,Y0,OPTIONS,P1,P2,...)

说明:此调用格式在 OPTIONS 中设置了一个函数 EVENTS,使其调用格式具有'Events'性质,在求如上的解的同时,还找(T,Y)的哪些函数值是零.在设定函数 EVENTS 时,才输出 TE,YE,IE.其中 TE 是事件发生时的自变量 t 构成的列向量,YE 的每一行是对应 TE 的数值解,向量 IE 指出哪一个事件发生.

以上用法可以通过下面的例题学习.

例 10.5.4 分别用二阶数值方法求初值问题 $\begin{cases}\dfrac{dy}{dx}=\dfrac{2x}{3y^2}, 0\leqslant x\leqslant 1.2,\\ y\big|_{x=0}=1\end{cases}$ 的数值解，精确到 10^{-4}，计算它与精确值的绝对误差和相对误差，并画出精确解和数值解的图形.

解 (1) 先求精确解. 输入程序

```
>>y=dsolve('(Dy)-2*x/(3*y^2)=0','y(0)=1','x')
```

运行后屏幕显示常微分方程在给定初始条件下的精确解 y 如下

```
y =
    (x^2+1)^(1/3)
```

(2) 用二阶龙格－库塔方法求初值问题的数值解，误差 10^{-4}，画出精确解和数值解的图形.

① 编写并保存名为 funfcn.m 的 MATLAB 程序如下

```
function f=funfcn(x,y)
f=2*x/(3*y^2);
```

② 在 MATLAB 工作窗口输入下面的程序

```
>>options=odeset('RelTol',1e-4,'AbsTol',1e-4);
[t,y]=ode23(@funfcn,[0 1.2],1,options),yf=(t.^2+1).^(1/3)
plot(t,y(:,1),'rp',t,yf,'bo');
grid,xlabel('自变量 X'),ylabel('因变量 Y')
legend('用二阶龙格-库塔方法计算 dy/dx=2x/(3y^2),y(0)=1 在[0,
1.2]上的数值解','dy/dx=2x/(3y^2),y(0)=1 的精确解 y=f(x)')
juew=yf(:,1)-y(:,1),
xiangw=juew./yf(:,1),[t,y,yf,juew,xiangw]
```

运行后计算的结果列入表 10－13，并画出精确解和数值解的图形(见图 10－17).

表 10－13 用二阶龙格－库塔方法求初值问题的结果

自变量 t	数值解 y	精确解 yf	$yf-y$	$(yf-y)/yf$
0	1.000 0	1.000 0	0	0
0.12	1.004 8	1.004 8	0.000 0	0.000 0
0.24	1.018 8	1.018 8	0.000 0	0.000 0
0.36	1.041 5	1.041 5	0.000 0	0.000 0
0.48	1.071 6	1.071 6	0.000 0	0.000 0
0.60	1.107 9	1.107 9	0.000 0	0.000 0
0.72	1.149 4	1.149 4	0.000 0	0.000 0

续表

自变量 t	数值解 y	精确解 yf	$yf-y$	$(yf-y)/yf$
0.84	1.194 8	1.194 8	0.000 0	0.000 0
0.96	1.243 2	1.243 2	0.000 0	0.000 0
1.08	1.293 9	1.293 9	0.000 0	0.000 0
1.20	1.346 3	1.346 3	0.000 0	0.000 0

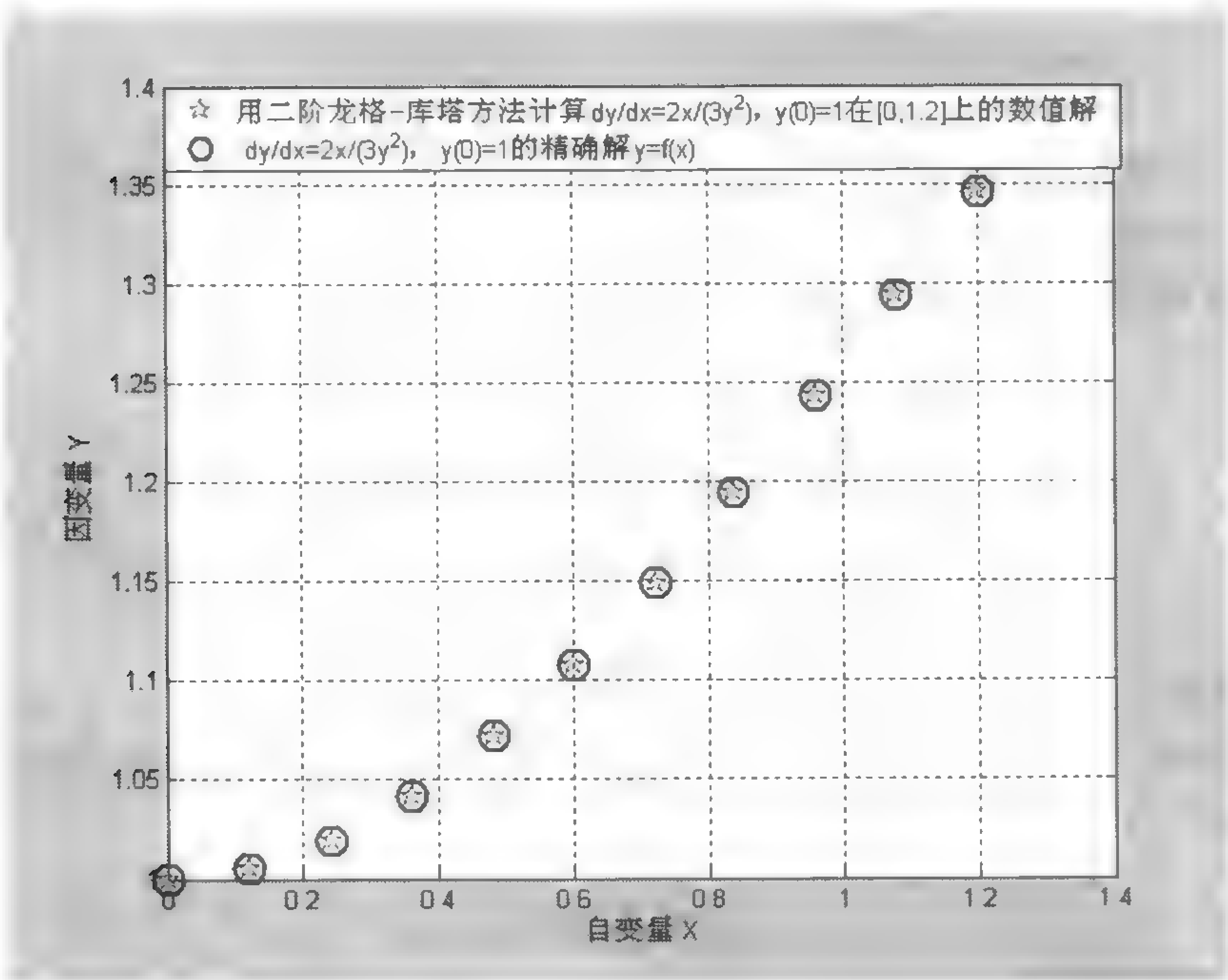


图 10－17 用二阶龙格－库塔方法计算 $dy/dx=2x/(3y^2)$，$y(0)=1$ 的数值解

通过上面的计算结果可见，用 ode23 求初值问题的数值解与精确值的绝对误差和相对误差达到了精确到 10^{-4}.

例 10.5.5 对例 10.5.4 也可以将 ode23 函数换成 ode45 函数求其初值问题的数值解，精确到 10^{-4}时得到的结果是相同的. 试对这两种方法进行比较.

解 编写名为 tode23.m 的 M 文件

```
function tode23
tic; p1 = flops;
options = odeset('RelTol', 1e-4,'AbsTol',1e-4);
[x,y] = ode23(@funfcn,[0 1.2],1,options);
```

```
p2 = flops; tode23 = toc,pode23 = p2 - p1
```

在 MATLAB 工作窗口输入文件名

```
>> tode23
```

运行后输出运行时间如下

```
tode23 =
          0.2800
```

把名为 tode23.m 的 M 文件中的 ode23 改成 ode45. 运行文件名

```
>> tode45
```

后,得

```
tode45 =
        0.2200
```

从此例所得结果可以看出,ode23 比 ode45 确实慢一些,但是 ode23 比欧拉方法要快.

例 10.5.6 先判别方程组

$$\begin{cases} \dfrac{dy}{dx} = -2y + z + 2\sin x, \\ \dfrac{dz}{dx} = 998y - 999z + 999(\cos x - \sin x), \\ y|_{x=0} = 2, \\ z|_{x=0} = 3 \end{cases}$$

是否是刚性的. 再分别用 ode45 函数和 ode15s 函数求在[0, 1.2]上的数值解,精确到 10^{-1},将计算结果与精确值比较,并画出精确解和数值解的图形.

解 (1) 在 MATLAB 中求刚性比. 输入程序

```
>> A = [-2 1;998 -999]; Tez1 = max(abs(real(eig(A))));
Tez2 = min(abs(real(eig(A)))); Gxb = Tez1/Tez2
```

运行后输出刚性比为

```
Gxb =
    1000
```

因为刚性比 1 000 远大于 1,所以原方程组为刚性方程组.

(2) 求精确解. 输入程序

```
>> syms x y z
f = '(Dy)  2 * sin(x) - z + 2 * y = 0,(Dz) + 999 * sin(x) - 999 * cos(x)
+ 999 * z - 998 * y = 0 ';
[y,z] - dsolve(f,'y(0) = 2,z(0) = 3 ','x')
```

运行后屏幕显示常微分方程组在给定初始条件下的精确解 y 和 z 如下

```
y =                                z =
       2 * exp( - x) + sin(x)      2 * exp( - x) + cos(x)
```

(3) 分别用 ode45 函数和 ode15s 函数解刚性方程的数值解,误差 10^{-1},画出精确解和数值解的图形.

① 编写并保存名为 funfcn.m 的 MATLAB 程序如下

```
function f = funfcn(x,u)
f = [-2 1;998  -999] * u + [2 * sin(x); -999 * sin(x) + 999 *
cos(x)];
```

② 在 MATLAB 工作窗口输入下面的程序

```
function li1056
options = odeset('RelTol', 1e-1,'AbsTol',[1e-1,1e-1]);
[t,y1] = ode15s(@funfcn,[0 10],[2 3],options)
yf = 2 * exp(-t) + sin(t); zf = 2 * exp(-t) + cos(t);
plot(t,y1(:,1),'mo',t,y1(:,2),'rp');
hold on,plot(t,yf,'b-',t,zf,'g-.'); hold off
grid,xlabel('自变量 X'), ylabel('因变量 Y')
legend('用 ode15s 函数解刚性方程的 y = f(x)数值解','用 ode15s 函数解
刚性方程的 z = g(x)数值解','精确解 y = f(x)','精确解 z = g(x)')
```

运行后屏幕显示计算的结果(略),并画出精确解和用 ode15s 求的数值解的图形(见图 10-18(a)).

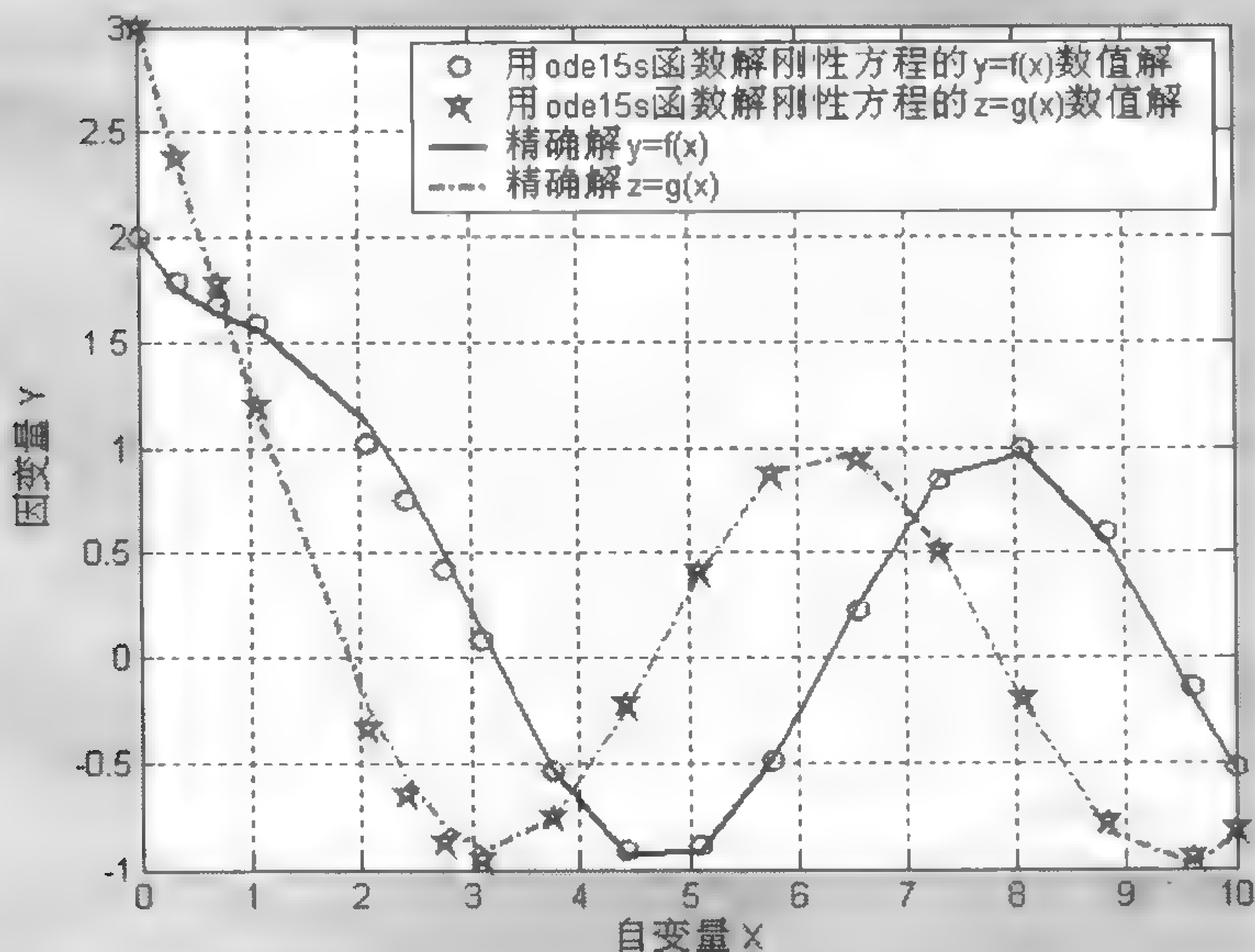


图 10-18(a) 精确解和用 ode15s 求数值解的图

将 ode15s 改为 ode45,则画出精确解和用 ode15s 求的数值解的图形(见图 10-18(b))

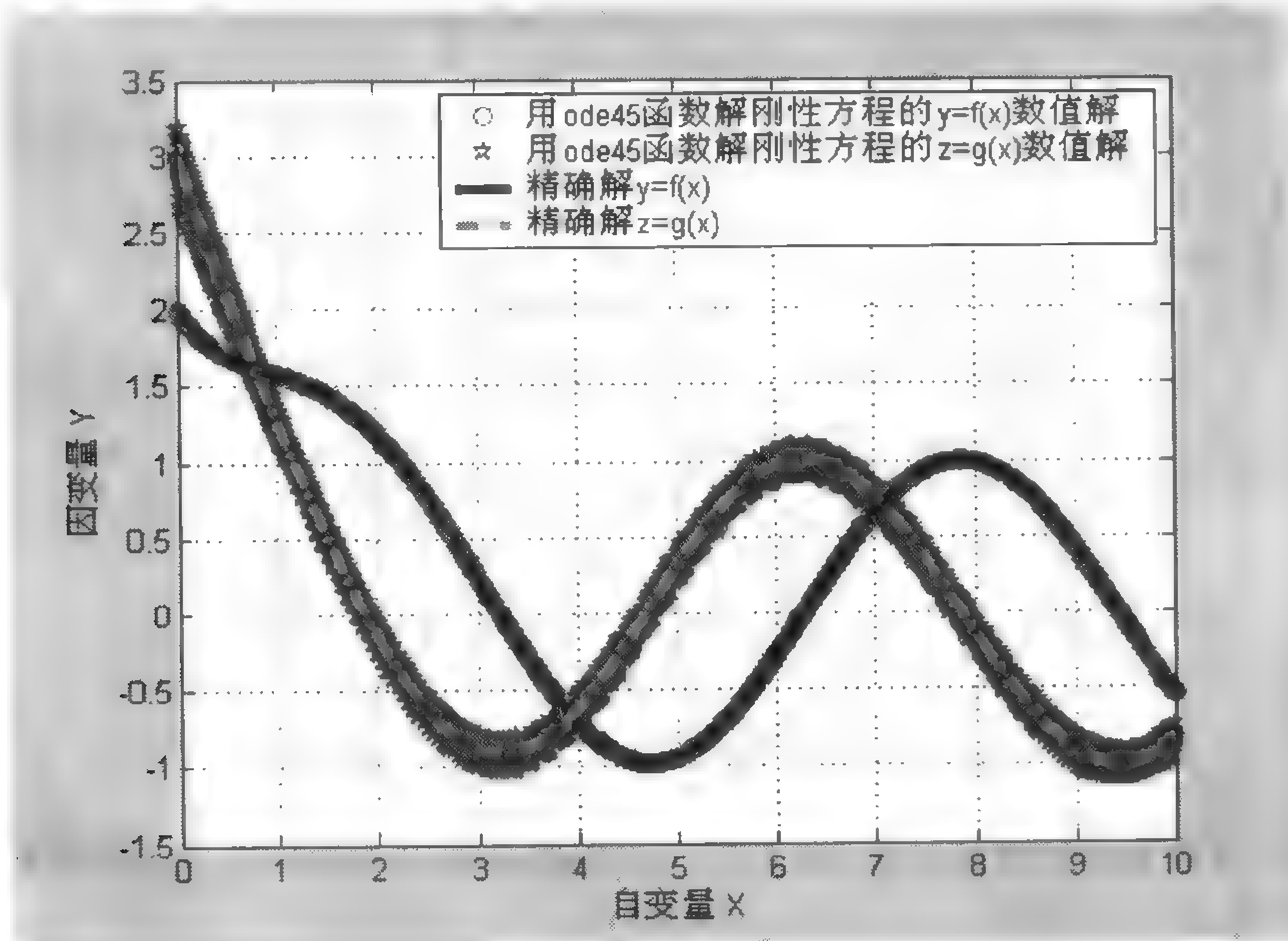


图 10-18(b) 精确解和用 ode45 求数值解的图

比较图 10-18(a)和图 10-18(b)可见,对于刚性的方程组,用 ode15s 求数值解比用 ode45 速度快地多得多.下面计算时间,编写并保存 tode.m 的程序

```
function tode
tic; p1 = flops;
options = odeset('RelTol', 1e-1,'AbsTol',[1e-1,1e-1]);
[t,y1] = ode45(@funfcn,[0 10],[2 3],options);
p2 = flops;tode45 = toc,pode45 = p2 - p1,tic;,p3 = flops;
options = odeset('RelTol', 1e-1,'AbsTol',[1e-1,1e-1]);
[t,y2] = ode15s(@funfcn,[0 10],[2 3],options);
p4 = flops; tode15s = toc,pode15s = p4 - p3
```

输入命令

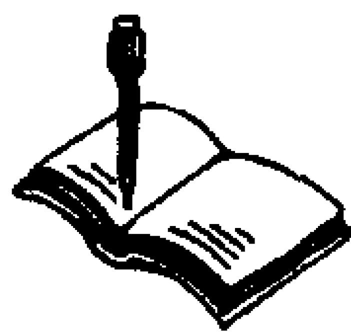
```
>> tode
```

运行后屏幕显示计算结果

```
tode45 =              tode15s =

        4.5100                  0.0500
```

由此可以看出,对于刚性的方程组,用 ode15s 求数值解比用 ode45 速度快许多.



习 题 10.5

1. 分别用二阶数值方法求下列初值问题

(1) $\begin{cases} \dfrac{dy}{dx}=y-\dfrac{2x}{y},0\leqslant x\leqslant 2, \\ y|_{x=0}=1; \end{cases}$　　(2) $\begin{cases} \dfrac{dy}{dx}=x+y,0\leqslant x\leqslant 1, \\ y|_{x=0}=1 \end{cases}$

的数值解，并计算与精确解的误差，画出精确解和数值解的图形.

2. 先判别方程组

$$\begin{cases} \dfrac{dy}{dx}=-3y+2z+\sin x, \\ \dfrac{dz}{dx}=998y-999z+999(\cos x-\sin x), \\ y|_{x=0}=2, \\ z|_{x=0}=3 \end{cases}$$

是否是刚性的. 再分别用 ode45 函数和 ode15s 函数求在[0, 1.2]上的数值解，精确到 10^{-1}，将计算结果与精确值比较，并画出精确解和数值解的图形.

3. 用表 10－12 中的库函数求下列初值问题

(1) $\begin{cases} \dfrac{dy}{dx}=\dfrac{3y}{1+x},0\leqslant x\leqslant 1, \\ y|_{x=0}=1; \end{cases}$　　(2) $\begin{cases} \dfrac{dy}{dx}=x^2+x^3y,0\leqslant x\leqslant 2, \\ y|_{x=1}=1 \end{cases}$

的数值解，并计算与精确解的误差，画出精确解和数值解的图形.

4. 选择表 10－12 中合适的库函数解初值问题

$$\begin{cases} \dfrac{dy}{dx}=-y+z+2\sin x, \\ \dfrac{dz}{dx}=998y-999z+999(\cos x-\sin x), \\ y|_{x=0}=2, \\ z|_{x=0}=3. \end{cases}$$

5. 选择表 10－12 中合适的库函数解计算积分 $I(x)=\int_0^x e^{t^2}dt$ 在 $x=1$ 附近的近似值，取 $h=0.2$，计算结果保留四位小数.

6. 选择表 10－12 中合适的库函数解下列初值问题

(1) $\begin{cases} (1+x^2)\dfrac{dy}{dx}+2xy=4x^2,1\leqslant x\leqslant 2, \\ y|_{x=0}=1; \end{cases}$　　(2) $\begin{cases} \dfrac{dy}{dx}=x^2+x^3y,0\leqslant x\leqslant 2, \\ y|_{x=1}=1 \end{cases}$

的数值解，要求精确到 10^{-6}，并计算与精确解的误差，画出精确解和数值解的图形.

7. 选择表 10－12 中合适的库函数解初值问题

$$\begin{cases}\dfrac{dy}{dx}=1-\dfrac{2ty}{1+t^2},0<t\leqslant 2,\\ y(0)=0.\end{cases}$$

取步长 $h=0.5$，小数点后至少保留六位.

8. 选择表 10－12 中合适的库函数求初值问题$\begin{cases}y'=-xy+\dfrac{4x}{y},0\leqslant x\leqslant 1,\\ y(0)=1;h=0.25\end{cases}$的数值解.

9. 选择表 10－12 中合适的库函数求下列初值问题的数值解.

(1) $\begin{cases}y'=-y+x+1,0\leqslant x\leqslant 1,\\ y(0)=1;h=0.1;\end{cases}$ (2) $\begin{cases}y'=\dfrac{1}{x}(y^2+y),1\leqslant x\leqslant 3,\\ y(0)=-2;h=0.5.\end{cases}$

10.6 线性多步法及其 MATLAB 程序

本章上述介绍的各种常微分方程初值问题

$$\frac{dy}{dx}=f(x,y),y(x_0)=y_0,$$

数值解法都是单步法.单步法的一般形式是

$$y_{n+1}=y_n+hg(x_n,y_n,h)(n=0,1,2,\cdots,N-1).$$

这是因为单步法在计算 y_{n+1}时都只用到前一步的值 y_n，为了提高精度，需重新计算多个点处的函数值(如龙格－库塔方法)，计算量较大.多步法的基本思想是如何通过较多地利用前面的已知信息(如 $y_n,y_{n-1},\cdots,y_{n-r}$)来构造高精度的算法计算 y_{n+1}.构造线性多步法公式常用泰勒展开法和数值积分法.

10.6.1 线性多步法一般形式及其截断误差

最常用的多步法是线性多步法，其一般形式为

$$y_{n+1}=\sum_{i=0}^{k-1}\alpha_i y_{n-i}+h\sum_{i=-1}^{k-1}\beta_i f_{n-i},n=k,k+1,\cdots, \tag{10.47}$$

其中 $f_{n-i}=f(x_{n-i},y_{n-i})$，(10.47)式中的 $2k+1$ 个待定常数 α_i,β_i满足

$$\begin{cases}\displaystyle\sum_{i=0}^{k-1}\alpha_i=1,\\ \displaystyle\sum_{i=1}^{k-1}(-i)^r\alpha_i+r\sum_{i=-1}^{k-1}(-i)^{r-1}\beta_i=1(r=1,2,\cdots,p).\end{cases} \tag{10.48}$$

(10.48)式解所对应的(10.47)式具有 p 阶精度，局部截断误差为

$$R_{n+1}=\frac{h^{p+1}}{(p+1)!}\Big[1-\sum_{i=1}^{k-1}(-i)^{p+1}\alpha_i-(p+1)\sum_{i=-1}^{k-1}(-i)^p\beta_i\Big]y_n^{(p+1)}+O(h^{p+2}). \tag{10.49}$$

线性多步法(10.47)式至多可达到 $2k$ 阶精度.

如果将(10.47)式右端 $\alpha_i^2+\beta_i^2\neq 0$,则称(10.47)为**线性 k 步法**,计算时用到前面已算出的 k 个导数值 $f_{n-k+1},\cdots,f_{n-1},f_n$. 当 $\beta_{-1}=0$ 时,(10.47)式右端是已知的,称(10.47)为**显式多步法**. 当 $\beta_{-1}\neq 0$ 时,(10.47)式右端未知的,$y'_{n+1}=f(x_{n+1},y_{n+1})$,称(10.47)为**隐式多步法**.

利用线性多步法求解常微分方程初值问题(10.5)时,必须先利用其他方法算出几个点的近似值,一般可以用同阶的单步法. 常用的线性多步法的公式有如下几种.

10.6.2 亚当斯(Adams)显式公式及其 MATLAB 程序

取 $k=4,\alpha_1=\alpha_2=\alpha_3=\beta_{-1}=0$,由方程组(10.48),可得

$$\alpha_0=1,\beta_0=\frac{55}{24},\beta_1=-\frac{59}{24},\beta_2=\frac{37}{24},\beta_3=-\frac{9}{24}$$

代入(10.47)式中,得**四阶亚当斯显式公式**,即**四阶亚当斯外插公式**为

$$y_{n+1}=y_n+\frac{h}{24}(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}), \tag{10.50}$$

局部截断误差为

$$\begin{aligned}R_{n+1}&=\frac{h^5}{5!}\Big[1-\sum_{i=1}^{3}(-i)^5\alpha_i-5\sum_{i=-1}^{5}(-i)^4\beta_i\Big]y_n^{(5)}+O(h^6)\\&=\frac{251}{720}h^5y_n^{(5)}+O(h^6).\end{aligned} \tag{10.51}$$

根据四阶亚当斯显式公式(10.50)式和常用的四阶龙格-库塔公式编写求解常微分方程初值问题(10.5)的 MATLAB 程序如下.

> **用四阶亚当斯显式公式求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序**
>
> 输入量:*funfcn* 是函数 $f(x,y)$,x_0 和 y_0 是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,h 是步长.
>
> 输出量:向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值,向量 $\boldsymbol{Y}$ 的元素是利用(10.50)式求出的初值问题(10.5)在向量 $\boldsymbol{X}$ 的元素处的数值解,n 是自变量 x 取值的序号,wucha(k+1)=norm(Y(k+1)-Y(k)),并画出数值解和精确解的图形.

```
function [k,X,Y,wucha,P] = Adams4x(funfcn,x0,b,y0,h)
x = x0; y = y0;p = 128; n = fix((b - x0)/h);
if n < 5,return,end;
X = zeros(p,1); Y = zeros(p,length(y)); f = zeros(p,1);
```

```
k=1; X(k)=x; Y(k,:)=y';
for k=2:4
c1=1/6;c2=2/6;c3=2/6;c4=1/6;a2=1/2; a3=1/2;
a4=1;b21=1/2;b31=0;b32=1/2; b41=0;b42=0;b43=1;
x1=x+a2*h; x2=x+a3*h; x3=x+a4*h; k1=feval(funfcn,x,y);
y1=y+b21*h*k1; x=x+h; k2=feval(funfcn,x1,y1);
y2=y+b31*h*k1+b32*h*k2; k3=feval(funfcn;x2,y2);
y3=y+b41*h*k1+b42*h*k2+b43*h*k3; k4=feval(funfcn,x3,y3);
y=y+h*(c1*k1+c2*k2+c3*k3+c4*k4); X(k)=x; Y(k,:)=y;
end
X;Y;f(1:4)=feval(funfcn,X(1:4),Y(1:4));
for k-4:n
f(k)=feval(funfcn,X(k),Y(k));
X(k+1)=X(1)+h*k; Y(k+1)=Y(k)+(h/24)*((f(k-3:k))'*
[-9 37 -59 55]');
f(k+1)= feval(funfcn,X(k+1),Y(k+1));f(k)-f(k+1); k=k+1;
end
for k=2:n+1
wucha(k)=norm(Y(k)-Y(k-1)); k=k+1;
end
X=X(1:n+1); Y=Y(1:n+1,:); n=1:n+1,
wucha=wucha(1:n,:); P=[n',X,Y,wucha'];
```

例 10.6.1 先利用常用的四阶龙格－库塔公式求初值问题

$$\begin{cases}\dfrac{dy}{dx}=1-\dfrac{2xy}{1+x^2},0\leqslant x\leqslant 2,\\ y|_{x=0}=0,\end{cases}$$

的几个点的数值解,再利用四阶亚当斯显式公式求解常微分方程初值问题,$h=\frac{1}{15}$,并计算它与精确解的误差,在同一图形窗口画出精确解和数值解的图形.

解 (1)对此问题的四阶亚当斯显式公式的具体形式为

$$\begin{cases}f_n=1-\dfrac{2x_ny_n}{1+x_n^2},0\leqslant x_n\leqslant 2,n=4,5,6,\cdots,\\ y_{n+1}=y_n+\dfrac{h}{24}(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}).\end{cases}$$

常用的四阶龙格－库塔公式的具体形式为

$$\begin{cases} x = x_0 + jh, x_0 = 0, j = 0,1,2,\cdots,\left[\dfrac{b - x_0}{h}\right], \\ y_{n+1} = y_n + h(c_1k_1 + c_2k_2 + c_3k_3 + c_4k_4), n = 0,1,2,3. \\ k_1 = 1 - \dfrac{2x_ny_n}{1 + x_n^2} \\ k_2 = 1 - \dfrac{2(x_n + a_2h)(y_n + b_{21}hk_1)}{1 + (x_n + a_2h)^2}, \\ k_3 = 1 - \dfrac{2(x_n + a_3h)(y_n + b_{31}hk_1 + b_{32}hk_2)}{1 + (x_n + a_3h)^2}, \\ k_4 = 1 - \dfrac{2(x_n + a_4h)(y_n + b_{41}hk_1 + b_{42}hk_2 + b_{43}hk_3)}{1 + (x_n + a_4h)^2}. \end{cases}$$

取 $b_{31} = b_{41} = b_{42} = 0, b_{43} = 1, c_1 = \dfrac{1}{6}, c_2 = \dfrac{1}{3}, c_3 = \dfrac{1}{3}, c_4 = \dfrac{1}{6}, a_4 = 1, a_2 = a_3 = b_{21} = b_{32} = \dfrac{1}{2}$.

(2) 编写并保存名为 funfcn.m 和 fun.m 的 M 文件如下

```
function f = funfcn(x,y)
f =1 -(2.*x.*y)./(1 +x.^2);
function y = fun (x)
y =(x +1/3.*x.^3)./(1 +x.^2);
```

(3) 在 MATLAB 工作窗口输入下面的程序

```
>> y =dsolve('Dy =1 -(2*x*y)/(1 +x^2)','x')
>> x0 -0;b -2;y0 -0;h -1/15;
[k,X,Y,wucha,P] =Adams4x(@funfcn,x0,b,y0,h),
y =(X +1/3*X.^3)./(1 +X.^2); b31 =0; b41 -0;b42 =0;b43 =1;
c1 =1/6;c2 =1/3;c3 =1/3;c4 =1/6;a2 =1/2;
a3 =1/2; a4 =1;b21 =1/2; b32 =1/2;
C =[c1,c2,c3, c4,a2, a3, a4,b21,b31,b32,b41,b42,b43];
[k,X,Y1,fxy,wch,wucha,P] =RK4(@funfcn,@fun,x0,b,C,y0,h)
plot(X,Y,'gh',X,Y1,'mp',X,y,'bo'),
grid,xlabel('自变量 X'),ylabel('因变量 Y')
legend('用四阶亚当斯显式公式计算 dy/dx =1 -(2xy)/(1 +x^2),y(0) =0
在[0,2]上的数值解','用常用的四阶龙格 -库塔公式计算 dy/dx =1 -(2xy)/(1 +x^
2),y(0) =0 在[0,2]上的数值解','dy/dx =1 -(2xy)/(1 +x^2)的精确解 y =(x +1/
3x^3)/(1 +x^2)')
wchY =abs(y -Y), wchY1 =abs(y -Y1), m =zeros(1,k),
for n -1:k
```

```
m(1,n)=n-1
end
[m',X,y,Y,Y1,wchY,wchY1]
```

运行后屏幕显示图 10 - 19 和计算结果(见表 10 - 14). 由此可见,四阶亚当斯显式公式在前 4 次迭代的数值解与精确解的绝对误差较小,但是从第 5 次迭代开始到第 30 次迭代为止,随着自变量的增大,四阶亚当斯显式公式计算的数值解与精确解的绝对误差逐渐增大,大约在 $x=1$ 左右又逐渐减少,但是误差始终比四阶龙格 - 库塔公式的误差大,尤其是在 $x_n=0.933\ 3$ 时,误差最大.

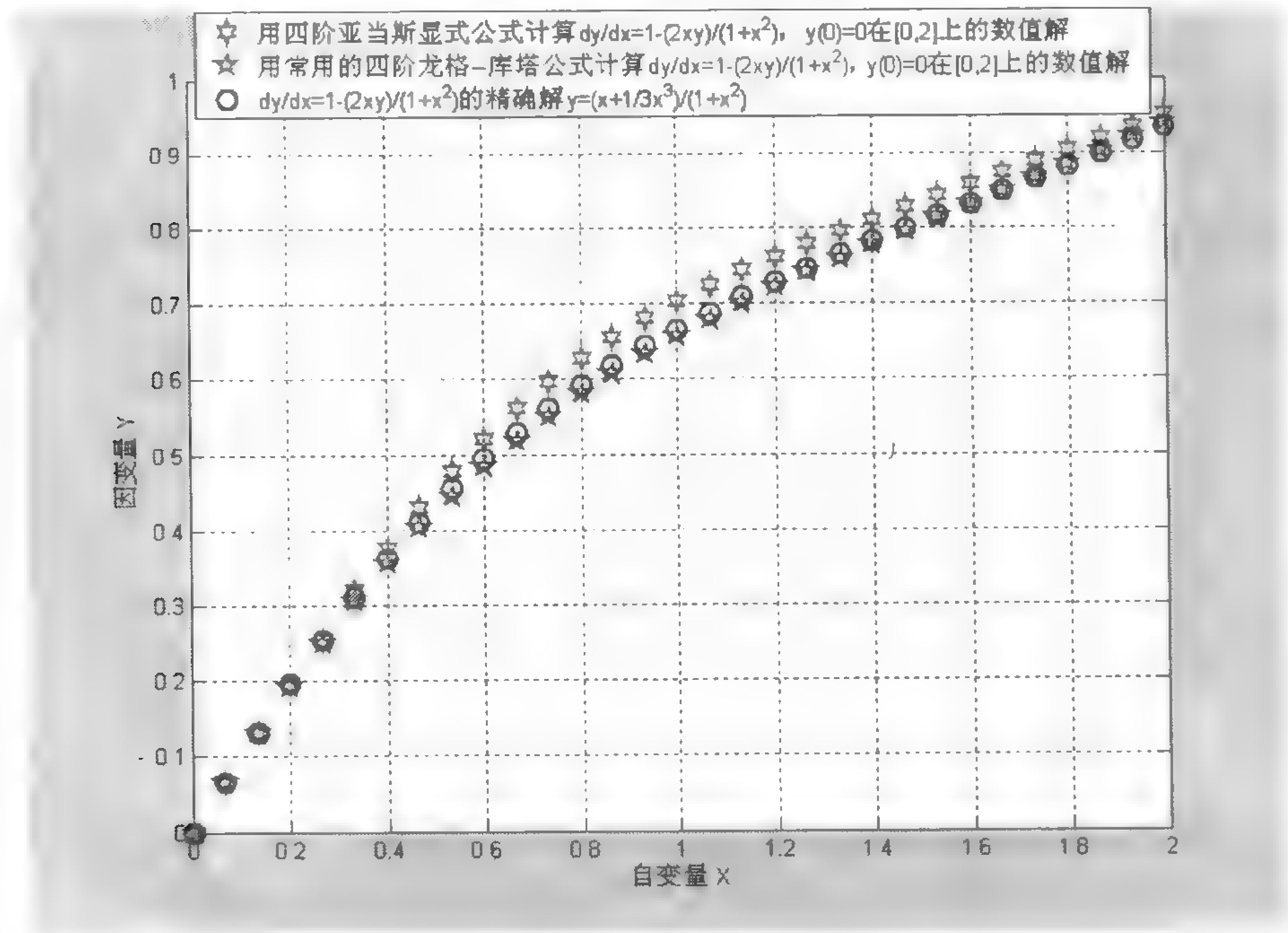


图 10 - 19 用四阶龙格 - 库塔公式和四阶亚当斯显式公式求解常微分方程初值问题

表 10 - 14 用四阶龙格 - 库塔公式和四阶亚当斯显式公式求解常微分方程初值问题

n	x_n	$y(x_n)$	Y	Y_1	$\|y(x_n)-Y\|$	$\|y(x_n)-Y_1\|$
0	0	0	0	0	0	0
1.000 0	0.066 7	0.066 5	0.066 5	0.066 2	0.000 0	0.000 3
2.000 0	0.133 3	0.131 8	0.131 8	0.130 7	0.000 0	0.001 1
3.000 0	0.200 0	0.194 9	0.194 9	0.192 5	0.000 0	0.002 3
4.000 0	0.266 7	0.254 9	0.254 8	0.251 0	0.000 0	0.003 8

续表

n	x_n	$y(x_n)$	Y	Y_1	$\|y(x_n)-Y\|$	$\|y(x_n)-Y_1\|$
5.000 0	0.333 3	0.311 1	0.319 6	0.305 7	0.008 5	0.005 4
6.000 0	0.400 0	0.363 2	0.376 2	0.356 2	0.013 0	0.007 0
7.000 0	0.466 7	0.411 0	0.429 2	0.402 6	0.018 2	0.008 4
8.000 0	0.533 3	0.454 6	0.477 7	0.445 0	0.023 1	0.009 6
9.000 0	0.600 0	0.494 1	0.521 4	0.483 7	0.027 3	0.010 4
10.000 0	0.666 7	0.529 9	0.560 7	0.518 9	0.030 7	0.011 0
11.000 0	0.733 3	0.562 4	0.595 7	0.551 2	0.033 3	0.011 2
12.000 0	0.800 0	0.591 9	0.627 0	0.580 7	0.035 1	0.011 2
13.000 0	0.866 7	0.618 8	0.655 0	0.608 0	0.036 2	0.010 9
14.000 0	0.933 3	0.643 7	0.680 2	0.633 3	0.036 6	0.010 4
15.000 0	1.000 0	0.666 7	0.703 1	0.657 0	0.036 4	0.009 7
16.000 0	1.066 7	0.688 2	0.724 0	0.679 3	0.035 8	0.008 9
17.000 0	1.133 3	0.708 5	0.743 4	0.700 5	0.034 9	0.008 0
18.000 0	1.200 0	0.727 9	0.761 6	0.720 8	0.033 7	0.007 0
19.000 0	1.266 7	0.746 5	0.778 8	0.740 4	0.032 4	0.006 0
20.000 0	1.333 3	0.764 4	0.795 4	0.759 4	0.030 9	0.005 0
21.000 0	1.400 0	0.782 0	0.811 4	0.778 0	0.029 4	0.004 0
22.000 0	1.466 7	0.799 2	0.827 0	0.796 2	0.027 9	0.003 0
23.000 0	1.533 3	0.816 2	0.842 5	0.814 2	0.026 3	0.001 9
24.000 0	1.600 0	0.833 0	0.857 8	0.832 0	0.024 8	0.001 0
25.000 0	1.666 7	0.849 7	0.873 0	0.849 7	0.023 3	0.000 0
26.000 0	1.733 3	0.866 3	0.888 3	0.867 3	0.021 9	0.000 9
27.000 0	1.800 0	0.883 0	0.903 6	0.884 8	0.020 6	0.001 8
28.000 0	1.866 7	0.899 7	0.919 0	0.902 4	0.019 3	0.002 7
29.000 0	1.933 3	0.916 5	0.934 5	0.920 0	0.018 0	0.003 5
30.000 0	2.000 0	0.933 3	0.950 2	0.937 6	0.016 9	0.004 3

10.6.3 亚当斯隐式公式及其 MATLAB 程序

(一) 亚当斯隐式公式及其 MATLAB 程序

取 $\alpha_1=\alpha_2=\alpha_3=\beta_3=0$，由方程组(10.48)，可得

$$\alpha_0=1,\beta_{-1}=\frac{9}{24},\beta_0=\frac{19}{24},\beta_1=-\frac{5}{24},\beta_2=\frac{1}{24},$$

代入(10.47)式中得到**四阶亚当斯隐式公式**，即**四阶亚当斯内插公式**为

$$y_{n+1}=y_n+\frac{h}{24}(9f_{n+1}+19f_n-5f_{n-1}+f_{n-2}), \tag{10.52}$$

局部截断误差为

$$R_{n+1}=-\frac{19}{720}h^5y_n^{(5)}+O(h^6). \tag{10.53}$$

一般地，将四阶亚当斯隐式公式(10.52)写成显式形式很困难，所以我们可以用四阶龙格－库塔公式计算 y_{n+1}，然后代入式(10.52)右端计算左端的 y_{n+1}.

例 10.6.2 先利用常用的四阶龙格－库塔公式求初值问题

$$\begin{cases}\dfrac{dy}{dx}=x-y,0\leqslant x\leqslant 1,\\ y|_{x=0}=0,\end{cases}$$

的几个点的数值解，再利用四阶亚当斯隐式公式求解常微分方程初值问题，$h=\frac{1}{10}$，并计算它与精确解 $y=x-1+e^{-x}$ 的误差，在同一图形窗口画出精确解和数值解的图形.

解 (1) 先根据初值 $x_0=0,y_0=0$，步长 $h=0.1$ 和对此问题的常用的四阶龙格－库塔公式(10.43)的具体形式

$$\begin{cases}k_1=x_n-y_n,\\ k_2=\left(x_n+\dfrac{h}{2}\right)-\left(y_n+h\dfrac{k_1}{2}\right),\\ k_3=\left(x_n+\dfrac{h}{2}\right)-\left(y_n+h\dfrac{k_2}{2}\right),\\ k_4=(x_n+h)-(y_n+hk_3),\\ \bar{y}_{n+1}=y_n+\dfrac{h}{6}(k_1+2k_2+2k_3+k_4),\end{cases}$$

计算起步值即 y_1,y_2.

(2) 然后再根据对此问题的四阶亚当斯隐式公式的具体形式为

$$f_n=x_n-y_n,0\leqslant x_n\leqslant 1,$$

$$y_{n+1}=y_n+\frac{h}{24}(9\bar{f}_{n+1}+19f_n-5f_{n-1}+f_{n-2}).$$

即

$$y_{n+1}=y_n+\frac{h}{24}[9(x_{n+1}-\bar{y}_{n+1})+19(x_n-y_n)-5(x_{n-1}-y_{n-1})+(x_{n-2}-y_{n-2})].$$

取 $h=\frac{1}{10}, x_n=kh(k=0,1,2,\cdots,10)$,得

$$y_{n+1}=\frac{1}{24.9}(22.1y_n+0.5y_{n-1}-0.1y_{n-2}+0.24k+0.12),$$

计算 $y_3,y_4,\cdots,y_{10}$.

(3) 编写并保存名为 Adams4y.m 的 MATLAB 程序

```
function [k,X,Y,wucha,P] = Adams4y(x0,b,y0,h)
x = x0; y = y0;p = 128; n = fix((b - x0)/h);
if n < 5,return,end;
X = zeros(p,1); Y = zeros(p,length(y)); f = zeros(p,1);
k = 1; X(k) = x; Y(k,:) = y';
for k = 2:3
x1 = x + h/2; x2 = x + h/2; x3 = x + h;k1 = x - y;
y1 = y + h * k1/2;x = x + h;k2 = x1 - y1;
y2 = y + h * k2/2;k3 = x2 - y2;y3 = y + h * k3;k4 = x3 - y3;
y = y + h * (k1 + 2 * k2 + 2 * k3 + k4)/6;
X(k) = x;Y(k,:) = y; k = k + 1;
end
X,Y,
for k = 3:n
X(k + 1) = X(1) + h * k;
Y(k + 1) = (1/24.9) * (0.24 * k + 0.12 + (Y(k - 2:k))' * [ - 0.1 0.5 22.1]'),
k = k + 1,
end
for k = 2:n + 1
wucha(k) = norm(Y(k) - Y(k - 1));
end
X = X(1:n + 1);Y = Y(1:n + 1,:);n = 1:n + 1,
wucha = wucha(1:n,:);P = [n',X,Y,wucha'];
```

(4) 编写并保存名为 funfcn.m 和 fun.m 的 M 文件如下

```
function f = funfcn(x,y)
    f = x - y;
function y = fun (x)
   y - x - 1 + exp( - x);
```

(5) 在 MATLAB 工作窗口输入下面的程序

```
>> x0 =0;b =1;y0 =0;h =1/10;[k,X,Y,wucha,P] = Adams4y (x0,b,y0,h)
y =X -1 +exp( -X); b31 =0; b41 =0;b42 =0;b43 =1; c1 =1/6;c2 =1/3;
c3 =1/3;c4 =1/6;a2 =1/2; a3 =1/2; a4 =1;b21 =1/2; b32 =1/2;
C =[c1,c2,c3, c4,a2, a3, a4,b21,b31,b32,b41,b42,b43];
[k,X,Y1,fxy,wch,wucha,P] =RK4(@funfcn,@fun,x0,b,C,y0,h)
plot(X,Y,'gh',X,Y1,'mp',X,y,'bo'),
grid,xlabel('自变量 X'),ylabel('因变量 Y')
legend('用四阶亚当斯隐式公式计算 dy/dx =x -y,y(0) =0 在[0,1]上的数值解','用常用的四阶龙格 - 库塔公式计算 dy/dx =x -y,y(0) =0 在[0,1]上的数值解','dy/dx =x -y,y(0) =0 的精确解 y =x -1 +exp( -x)')
wchY =abs(y -Y), wchY1 =abs(y -Y1), m =zeros(1,k),
for n =1:k,m(1,n) =n -1,end, [m',X,y,Y,Y1,wchY,wchY1],
```

运行后屏幕显示图 10 - 20 和计算结果,见表 10 - 15.

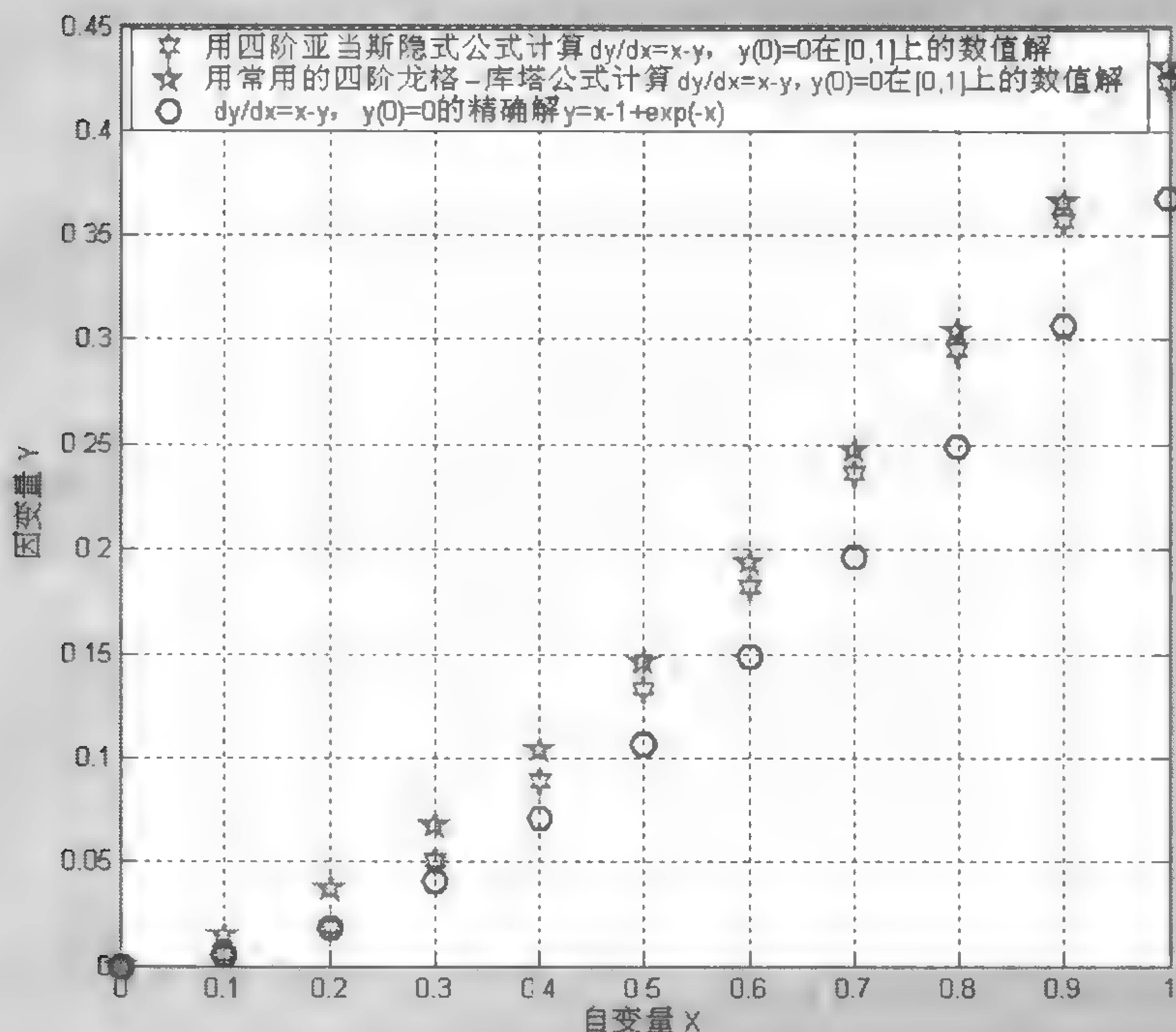


图 10 - 20 用四阶龙格 - 库塔公式和四阶亚当斯隐式公式求解常微分方程初值问题

表 10－15 用的四阶龙格－库塔公式和四阶亚当斯隐式公式求解常微分方程初值问题

n	x_n	$y(x_n)$	Y	Y_1	$\|y(x_n)-Y\|$	$\|y(x_n)-Y_1\|$
0	0	0	0	0	0	0
1.000 0	0.100 0	0.004 8	0.004 8	0.014 4	0.000 0	0.009 5
2.000 0	0.200 0	0.018 7	0.018 7	0.036 9	0.000 0	0.018 1
3.000 0	0.300 0	0.040 8	0.050 5	0.066 7	0.009 6	0.025 9
4.000 0	0.400 0	0.070 3	0.088 5	0.103 3	0.018 2	0.033 0
5.000 0	0.500 0	0.106 5	0.132 5	0.145 9	0.026 0	0.039 3
6.000 0	0.600 0	0.148 8	0.181 8	0.193 9	0.033 0	0.045 1
7.000 0	0.700 0	0.196 6	0.236 0	0.246 9	0.039 4	0.050 3
8.000 0	0.800 0	0.249 3	0.294 5	0.304 4	0.045 2	0.055 1
9.000 0	0.900 0	0.306 6	0.357 0	0.365 9	0.050 4	0.059 3
10.000 0	1.000 0	0.367 9	0.423 0	0.431 1	0.055 1	0.063 2

（二）改进的亚当斯方法及其 MATLAB 程序

从图 10－20 和表 10－15 可见，从 $x=0.3$ 开始，随着自变量的增大，四阶亚当斯隐式公式和四阶龙格－库塔公式计算的数值解分别与精确解的绝对误差也随之逐渐增大，但是前者比后者的误差较小些. 为了减少误差，提高精度，可对四阶亚当斯隐式公式（10.52）作改进：首先用常用的四阶龙格－库塔公式（10.43）计算常微分方程初值问题（10.5）的数值解 $\tilde{y}_i$，然后取 $y'_{i+1}=f(x_i,\tilde{y}_i)$ 代入公式（10.52）中求数值解 $y_i(i=1,2,\cdots)$. 根据这个思想，编写用改进的四阶亚当斯方法求解初值问题的 MATLAB 程序如下.

用改进的亚当斯方法求解常微分方程初值问题（10.5）的数值解的 MATLAB 主程序

输入量：*funfcn* 是函数 $f(x,y)$，x_0 和 y_0 是初值 $y(x_0)=y_0$，b 是自变量 x 的最大值，h 是步长.

输出量：向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值，向量 $\boldsymbol{Y}$ 的元素是利用（10.52）式求出初值问题（10.5）在向量 $\boldsymbol{X}$ 的元素处的数值解，n 是自变量 x 取值的序号，wucha(k+1)＝norm(Y(k+1)－Y(k))，并画出数值解和精确解的图形.

```
function [k,X,Y,wucha,P] = Adams4y1(funfcn,x0,b,y0,h)
x = x0;y = y0;p = 128; n = fix((b - x0)/h);
```

```
if n<5,return,end;
X=zeros(p,1); Y=zeros(p,length(y));
f=zeros(p,1);k=1; X(k)=x; Y(k,:)=y';
for k=2:4
c1=1/6;c2=2/6;c3=2/6;c4=1/6;a2=1/2; a3=1/2;a4=1;
b21=1/2;b31=0;b32=1/2; b41=0;b42=0;b43=1;x1=x+a2*h;
x2=x+a3*h; x3-x+a4*h;
k1=feval(funfcn,x,y);x=x+h;
y1=y+b21*h*k1;k2=feval(funfcn,x1,y1);
y2=y+b31*h*k1+b32*h*k2;k3=feval(funfcn,x2,y2);
y3=y+b41*h*k1+b42*h*k2+b43*h*k3;k4=feval(funfcn,x3,y3);
y=y+h*(c1*k1+c2*k2+c3*k3+c4*k4);X(k)=x;Y(k,:)=y;k=k+1;
end
X;Y;
% f(1:4)=feval(funfcn,X(1:4),Y(1:4));
for k=4:n
X(k+1)=X(1)+h*k;f(k+1)=feval(funfcn,X(k),Y(k));
Y(k+1)-Y(k)+(h/24)*((f(k-2:k+1))'*[1 -5 19 9]');
f(k+1)= feval(funfcn,X(k+1),Y(k+1));f(k)=f(k+1); k=k+1;
end
for k=2:n+1
wucha(k)-norm(Y(k)-Y(k-1));
end
X=X(1:n+1);Y=Y(1:n+1,:);n=1:n+1,
wucha=wucha(1:n,:);P=[n',X,Y,wucha'];
```

例 10.6.3 用改进的亚当斯方法和常用的四阶龙格-库塔公式求例10.6.2中初值问题的数值解及其误差，在同一图形窗口画出精确解和数值解的图形，并与例 10.6.2 的图形比较.

解 在 MATLAB 工作窗口输入下面的程序

```
>> x0=0;b=1;y0=0;h=1/10; [k,X,Y,wucha,P]=Adams4y1(@funf-
cn,x0,b,y0,h),
  y=X-1+exp(-X); b31=0; b41=0;b42=0;b43=1; c1=1/6;c2=2/6;
  c3=2/6;c4=1/6;a2=1/2; a3=1/2; a4=1;b21=1/2; b32=1/2;
  C=[c1,c2,c3, c4,a2, a3, a4,b21,b31,b32,b41,b42,b43];
  [k,X,Y1,fxy,wch,wucha,P]=RK4(@funfcn,@fun,x0,b,C,y0,h)
  plot(X,Y,'gh',X,Y1,'mp',X,y,'ro'),
  grid,xlabel('自变量 X'),ylabel('因变量 Y')
  legend('用改进的亚当斯方法计算 dy/dx=x-y,y(0)=0 在[0,1]上的数值
```

```
解','用常用的四阶龙格 - 库塔公式计算 dy/dx = x - y,y(0) = 0 在[0,1]上的数值解',
'dy/dx = x - y,y(0) = 0 的精确解 y = x - 1 + exp( - x)')
        wchY = abs(y - Y), wchY1 = abs(y - Y1), m = zeros(1,k),
        for n = 1:k,m(1,n) = n,end, [m',X,y,Y,Y1,wchY,wchY1],
```

运行后屏幕显示图 10 - 21 和计算结果(见表 10 - 16).由此可见,用改进的亚当斯方法比用四阶龙格 - 库塔公式计算的数值解与精确解的绝对误差小很多.比较图 10 - 21 和图 10 - 20 可以看出,改进的亚当斯方法比四阶亚当斯隐式公式计算的数值解的精度有显著的提高.

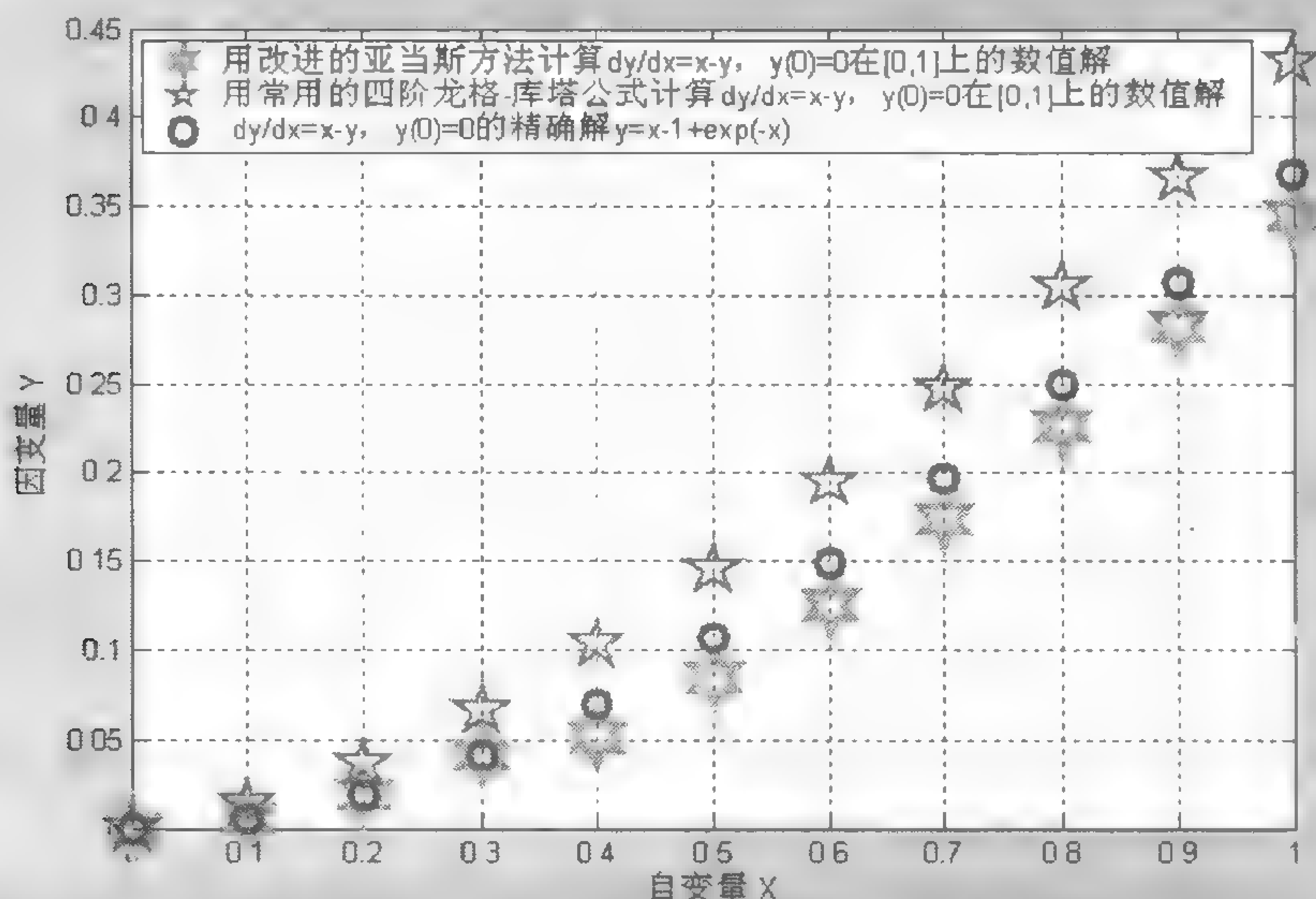


图 10 - 21 用改进的亚当斯方法求解常微分方程初值问题

表 10 - 16 用改进的亚当斯方法求解常微分方程初值问题

n	x_n	$y(x_n)$	Y_n	Y_{1n}	$\|y(x_n)-Y_n\|$	$\|y(x_n)-Y_{1n}\|$
0	0	0	0	0	0	0
1	0.100 0	0.004 8	0.004 8	0.014 4	0.000 0	0.009 5
2	0.200 0	0.018 7	0.018 7	0.036 9	0.000 0	0.018 1
3	0.300 0	0.040 8	0.040 8	0.066 7	0.000 0	0.025 9
4	0.400 0	0.070 3	0.067 7	0.103 3	0.002 6	0.033 0
5	0.500 0	0.106 5	0.100 3	0.145 9	0.006 3	0.039 3
6	0.600 0	0.148 8	0.140 0	0.193 9	0.008 8	0.045 1

续表

n	x_n	$y(x_n)$	Y_n	Y_{1n}	$\|y(x_n)-Y_n\|$	$\|y(x_n)-Y_{1n}\|$
7	0.700 0	0.196 6	0.185 7	0.246 9	0.010 9	0.050 3
8	0.800 0	0.249 3	0.236 9	0.304 4	0.012 4	0.055 1
9	0.900 0	0.306 6	0.293 0	0.365 9	0.013 5	0.059 3
10	1.000 0	0.367 9	0.353 5	0.431 1	0.014 3	0.063 2

10.6.4 米尔恩(Milne)公式及其 MATLAB 程序

取 $k=4,\alpha_0=\alpha_1=\alpha_2=\beta_{-1}=0$,由方程组(10.48),可得

$$\alpha_3=1,\beta_0=\frac{8}{3},\beta_1=-\frac{4}{3},\beta_2=\frac{8}{3},\beta_3=0$$

代入(10.47)式中得,米尔恩公式为

$$y_{n+1}=y_{n-3}+\frac{4}{3}h(2f_n-f_{n-1}+2f_{n-2}), \tag{10.54a}$$

局部截断误差为

$$R_{n+1}=\frac{14}{45}h^5y_n^{(5)}+O(h^6). \tag{10.55a}$$

米尔恩公式是四阶四步显式公式.

根据米尔恩公式(10.54a)式和常用的四阶龙格-库塔公式编写的求解常微分方程初值问题(10.5)的 MATLAB 程序如下.

> **用米尔恩公式求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序**
>
> 输入量:*funfcn* 是函数 $f(x,y)$,x_0和 y_0是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,h 是步长.
>
> 输出量:向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值,向量 $\boldsymbol{Y}$ 的元素是利用(10.54a)式求出初值问题(10.5)在向量 $\boldsymbol{X}$ 的元素处的数值解,n 是自变量 x 取值的序号,wucha(k+1)=norm(Y(k+1)-Y(k)),并画出数值解和精确解的图形.

```
function [k,X,Y,wucha,P] = Milne(funfcn,x0,b,y0,h)
x = x0;y = y0;p = 128; n = fix((b - x0)/h);
if n < 5,return,end;
X = zeros(p,1); Y = zeros(p,length(y));
f = zeros(p,1);k = 1; X(k) = x; Y(k,:) = y';
for k = 2:4
```

```
x = x + h; c1 = 1/6;c2 = 1/3;c3 = 1/3;c4 = 1/6;
a2 = 1/2; a3 = 1/2;a4 = 1;b21 = 1/2;b31 = 0;
b32 = 1/2; b41 = 0;b42 = 0;b43 = 1;x1 = x + a2 * h; x2 = x + a3 * h; x3 = x +
a4 * h;
k1 = feval(funfcn,x,y); y1 = y + b21 * h * k1;k2 = feval(funfcn,x1,y1);
y2 = y + b31 * h * k1 + b32 * h * k2;k3 = feval(funfcn,x2,y2);
y3 = y + b41 * h * k1 + b42 * h * k2 + b43 * h * k3;k4 = feval(funfcn,x3,y3);
y = y + h * (c1 * k1 + c2 * k2 + c3 * k3 + c4 * k4);X(k) = x;Y(k,:) = y;k = k + 1;
end
X;Y; f(1:4) = feval(funfcn,X(1:4),Y(1:4));
for k = 4:n
f(k) = feval(funfcn,X(k),Y(k));
X(k + 1) = X(1) + h * k;Y(k + 1) = Y(k - 3) + (4 * h/3) * ((f(k - 2:k))' *
[2 -1 2]');
f(k + 1) = feval(funfcn,X(k + 1),Y(k + 1)); f(k) = f(k + 1); k = k + 1;
end
for k = 2:n + 1
wucha(k) - norm(Y(k) - Y(k - 1));
end
X = X(1:n + 1);Y = Y(1:n + 1,:);n = 1:n + 1,
wucha = wucha(1:n,:);P = [n',X,Y,wucha'];
```

例 10.6.4 先利用常用的四阶龙格－库塔公式求初值问题$\frac{dy}{dx} = x - y, 0 \leqslant x \leqslant 4, y|_{x=0} = 0$ 的几个数值解，再分别利用米尔恩公式和改进的亚当斯方法及常用的四阶龙格－库塔公式求解其余的数值解，$h = 1/4$，并计算它们与精确解的误差，在同一图形窗口画出精确解和数值解的图形.

解 (1) 对此问题的米尔恩公式的具体形式为

$$\begin{cases} f_n = x_n - y_n, 0 \leqslant x_n \leqslant 4, n = 4,5,6,\cdots, \\ y_{n+1} = y_{n-3} + \frac{4h}{3}(2f_n - f_{n-1} + 2f_{n-2}). \end{cases}$$

(2) 编写并保存名为 funfcn.m 和 fun.m 的 M 文件如下

```
function f = funfcn(x,y)
    f = x - y;
function y = fun (x)
   y = x - 1 + exp( - x);
```

(3) 在 MATLAB 工作窗口输入下面的程序

```
>> x0 = 0;b = 4;y0 = 0;h = 1/4;
[k,X,Y,wucha,P] = Milne(@ funfcn,x0,b,y0,h)
```

```
[k,X,Y1,wucha,P] = Adams4y1(@ funfcn,x0,b,y0,h),
y = X - 1 + exp( - X); b31 = 0; b41 = 0;b42 = 0;b43 = 1;c1 = 1/6;c2 = 1/3;
c3 = 1/3;c4 = 1/6;a2 = 1/2; a3 = 1/2; a4 = 1;b21 = 1/2; b32 = 1/2;
C = [c1,c2,c3, c4,a2, a3, a4,b21,b31,b32,b41,b42,b43];
[k,X,Y2,fxy,wch,wucha,P] = RK4(@ funfcn,@ fun,x0,b,C,y0,h)
plot(X,Y,'gh',X,Y1,'mp',X,Y2,'ro',X,y,'b -'),
grid,xlabel('自变量 X'), ylabel('因变量 Y')
legend('用米尔恩公式计算 dy/dx = x - y,y(0) = 0 在[0,4]上的数值解','用改进的亚当斯方法计算 dy/dx = x - y,y(0) = 0 在[0,4]上的数值解','用常用的四阶龙格 - 库塔公式计算 dy/dx = x - y,y(0) = 0 在[0,4]上的数值解',' dy/dx = x - y 的精确解 y = x - 1 + exp( - x)')
wchY = abs(y - Y), wchY1 = abs(y - Y1), wchY2 = abs(y - Y2), m = zeros(1,k),
for n = 1:k,m(1,n) = n - 1,end, [m',X,y,Y,Y1,Y2,wchY,wchY1,wchY2],
```

运行后屏幕显示图 10 - 22 和计算结果(见表 10 - 17).

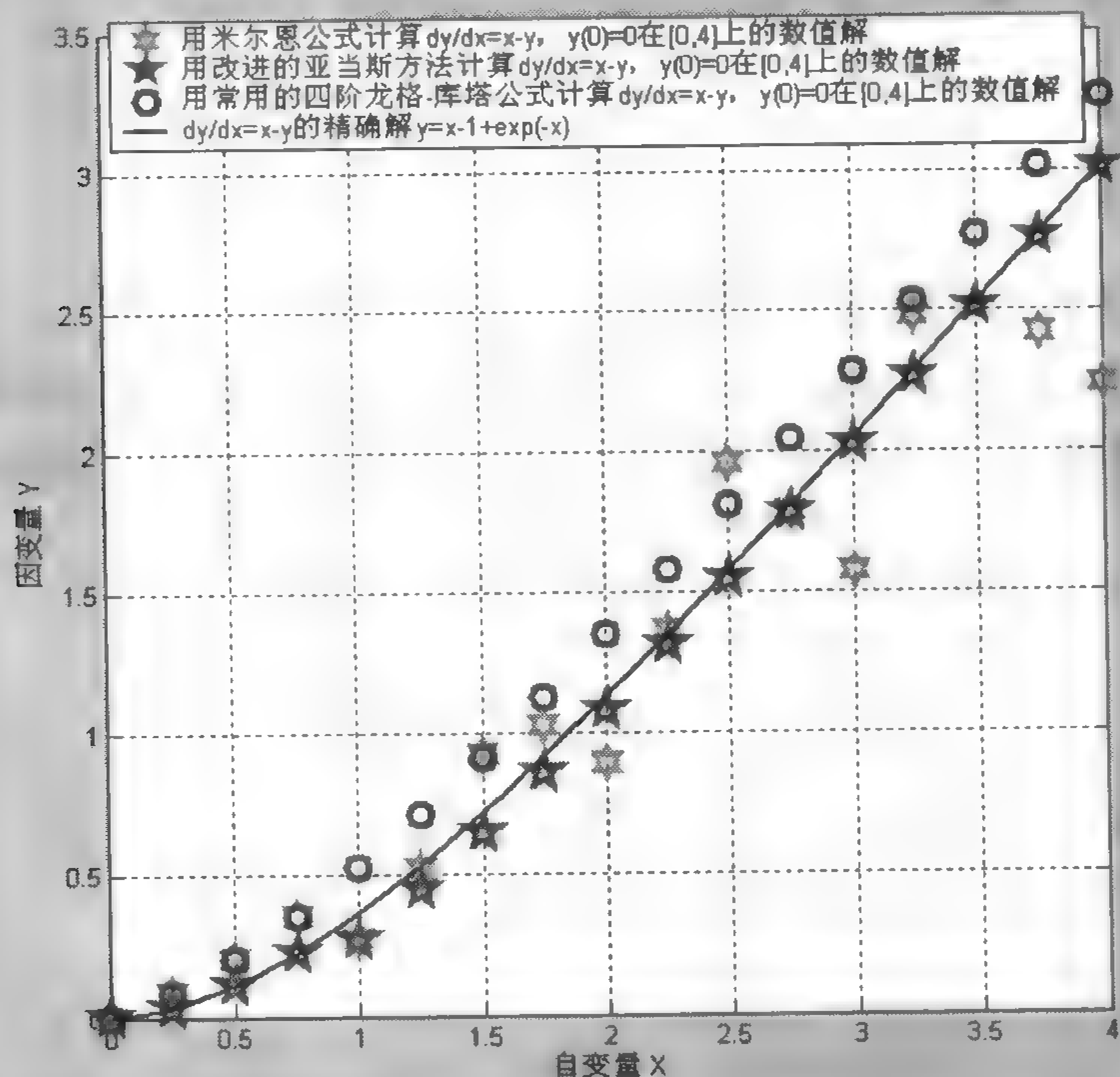


图 10 - 22 米尔恩公式、改进的亚当斯方法及常用的四阶龙格 - 库塔公式求数值解

表 10-17 用米尔恩公式、改进的亚当斯方法及四阶龙格-库塔公式求解常微分方程初值问题

n	x_n	$y(x_n)$	米尔恩 Y_n	改进的亚当斯 Y_{1n}	龙格-库塔 Y_{2n}	$\|y(x_n)-Y_n\|$	$\|y(x_n)-Y_{1n}\|$	$\|y(x_n)-Y_{2n}\|$
1	0.2500	0.0288	0.0841	0.0288	0.0841	0.0553	0.0000	0.0553
2	0.5000	0.1065	0.2049	0.1065	0.2049	0.0984	0.0000	0.0984
3	0.7500	0.2224	0.3543	0.2224	0.3543	0.1319	0.0000	0.1319
4	1.0000	0.3679	0.2760	0.2718	0.5259	0.0918	0.0960	0.1580
5	1.2500	0.5365	0.5222	0.4463	0.7149	0.0144	0.0902	0.1784
6	1.5000	0.7231	0.9302	0.6464	0.9174	0.2070	0.0767	0.1942
7	1.7500	0.9238	1.0295	0.8593	1.1303	0.1057	0.0645	0.2066
8	2.0000	1.1353	0.8961	1.0816	1.3515	0.2392	0.0537	0.2162
9	2.2500	1.3554	1.3705	1.3109	1.5791	0.0151	0.0445	0.2237
10	2.5000	1.5821	1.9593	1.5455	1.8116	0.3772	0.0366	0.2295
11	2.7500	1.8139	1.7961	1.7839	2.0480	0.0179	0.0300	0.2340
12	3.0000	2.0498	1.5746	2.0253	2.2873	0.4752	0.0245	0.2376
13	3.2500	2.2888	2.4816	2.2689	2.5291	0.1928	0.0199	0.2403
14	3.5000	2.5302	3.1657	2.5141	2.7727	0.6355	0.0161	0.2425
15	3.7500	2.7735	2.4198	2.7605	3.0176	0.3537	0.0130	0.2441
16	4.0000	3.0183	2.2409	3.0079	3.2637	0.7774	0.0105	0.2454

从图 10-22 和表 10-17 可以看出：改进的亚当斯方法计算的数值解 Y_{1n} 与精确解 $y(x_n)$ 的绝对误差 $|y(x_n)-Y_{1n}|$ 最小，常用的四阶龙格-库塔公式的绝对误差 $|y(x_n)-Y_{2n}|$ 次之，而米尔恩公式计算的数值解序列 $\{Y_n\}$ 呈现发散趋势，稳定性差，迭代 16 次时的绝对误差 $|y(x_n)-Y_n|=0.7774$，计算的结果 Y_n 已经毫无意义. 因此，人们常用公式

$$y_{n+1}=y_{n-1}+\frac{h}{3}[f(x_{n+1})+4f(x_n)+f(x_{n-1})] \tag{10.54b}$$

做校正，使其局部截断误差为

$$R_{n+1}=-\frac{1}{90}h^5y_n^{(5)}+O(h^6). \tag{10.55b}$$

这种方法称为**改进的米尔恩方法**，现提供用此方法求解常微分方程初值问题

(10.5)的 MATLAB 程序如下.

用改进的米尔恩方法求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序

输入量:*funfcn* 是函数 $f(x,y)$,x_0 和 y_0 是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,h 是步长.

输出量:向量 X 的元素是自变量 x 的取值,向量 Y 的元素是利用(10.54b)式求出的初值问题(10.5)在向量 X 的元素处的数值解,n 是自变量 x 取值的序号,wucha(k+1)=norm(Y(k+1)-Y(k)),并画出数值解和精确解的图形.

```
function [k,X,Y,wucha,P] = Milne1(funfcn,x0,b,y0,h)
x = x0;y = y0;p = 128; n = fix((b - x0) /h);
if n < 5,return,end;
X = zeros(p,1); Y = zeros(p,length(y));
f = zeros(p,1);k = 1; X(k) = x; Y(k,:) = y';
for k = 2:4
x = x + h; c1 = 1 /6;c2 = 1 /3;c3 = 1 /3;c4 = 1 /6;
a2 = 1 /2; a3 = 1 /2;a4 = 1;b21 = 1 /2;b31 = 0;
b32 = 1 /2; b41 = 0;b42 = 0;b43 = 1;x1 = x + a2 * h;
x2 = x + a3 * h; x3 = x + a4 * h;
k1 = feval(funfcn,x,y); y1 = y + b21 * h * k1;k2 = feval(funfcn,x1,y1);
y2 = y + b31 * h * k1 + b32 * h * k2;k3 = feval(funfcn,x2,y2);
y3 = y + b41 * h * k1 + b42 * h * k2 + b43 * h * k3;k4 = feval(funfcn,x3,y3);
y = y + h * (c1 * k1 + c2 * k2 + c3 * k3 + c4 * k4);X(k) = x;Y(k,:) = y;k = k + 1;
end
X;Y; f(1:4) = feval(funfcn,X(1:4),Y(1:4));
for k = 4:n
X(k + 1) = X(1) + h * k;f(k) = feval(funfcn,X(k),Y(k));
Y(k + 1) = Y(k - 3) + (4 * h/3) * ((f(k - 2:k))' * [2  -1 2]');
f(k + 1) =  feval(funfcn,X(k + 1),Y(k + 1));
Y1(k + 1) = Y(k - 1) + (h/3) * ((f(k - 1:k + 1))' * [1 4 1]');
f(k + 1) =  feval(funfcn,X(k + 1),Y1(k + 1));
Y(k) = Y1(k + 1);f(k) = f(k + 1); k = k + 1;
end
for k = 2:n + 1
wucha(k) = norm(Y(k) - Y(k - 1));
end
X = X(1:n + 1);Y = Y(1:n + 1,:);n = 1:n + 1,
```

```
wucha = wucha(1:n,:);P = [n',X,Y,wucha'];
```

例 10.6.5 先利用常用的四阶龙格－库塔公式求初值问题$\frac{dy}{dx}=x-y,0\leq x\leq 8,y|_{x=0}=0$的几个数值解，再分别利用**米尔恩方法**、米尔恩公式和改进的亚当斯方法及常用的四阶龙格－库塔公式求解其余的数值解，$h=\frac{1}{4}$，并计算它们与精确解的误差，在同一图形窗口画出精确解和数值解的图形.

解 (1) 编写并保存名为 funfcn.m 和 fun.m 的 M 文件如下

```
function f = funfcn1(x,y)
    f = x - y;
function y - fun1(x)
   y = x - 1 + exp( - x);
```

(2) 在 MATLAB 工作窗口输入下面的程序

```
>> x0 = 0;b = 8;y0 = 0;h = 1/4;
[k,X,Y,wucha,P] = Milne(@funfcn1,x0,b,y0,h)
[k,X,Yf,wucha,P] = Milne1(@funfcn1,x0,b,y0,h)
[k,X,Y1,wucha,P] = Adams4y1(@funfcn1,x0,b,y0,h),
c1 = 1/6;c2 = 1/3;
y = X  1 + exp( - X); b31 = 0; b41 = 0;b42 = 0;b43 = 1;
c3 = 1/3;c4 = 1/6;a2 = 1/2; a3 = 1/2;
a4 = 1;b21 = 1/2; b32 = 1/2;
C = [c1,c2,c3, c4,a2, a3, a4,b21,b31,b32,b41,b42,b43];
[k,X,Y2,fxy,wch,wucha,P] = RK4(@funfcn1,@fun1,x0,b,C,y0,h)
plot(X,Y,'mh',X,Yf,'b>',X,Y1,'gp',X,Y2,'ro',X,y,'b-'),
grid,xlabel('自变量 X'), ylabel('因变量 Y')
legend('用米尔恩公式计算 dy/dx = x - y,y(0) = 0 在[0,8]上的数值解','用
米尔恩方法计算 dy/dx = x - y,y(0) = 0 在[0,8]上的数值解','用改进的亚当斯方法计
算 dy/dx = x - y,y(0) = 0 在[0,8]上的数值解','用常用的四阶龙格 - 库塔公式计算
dy/dx - x - y,y(0) - 0 在[0,8]上的数值解','dy/dx = x - y 的精确解 y = x - 1 + exp
( - x)')
wchY = abs(y - Y),wchYf = abs(y - Yf),
wchY1 = abs(y - Y1),
wchY2 = abs(y - Y2), m = zeros(1,k),
for n = 1:k,m(1,n) = n - 1,end,
[m',X,y,Y,Yf,Y1,Y2,wchY,wchYf,wchY1,wchY2],
```

运行后屏幕显示图 10－23 和计算结果(略). 从图 10－23 可见，用米尔恩公式求出的数值解序列发散，但是修正后的米尔恩方法求出的数值解序列不但收敛，而且计算出的数值解与精确解的绝对误差比其他两种方法计算的误差

大不了多少.

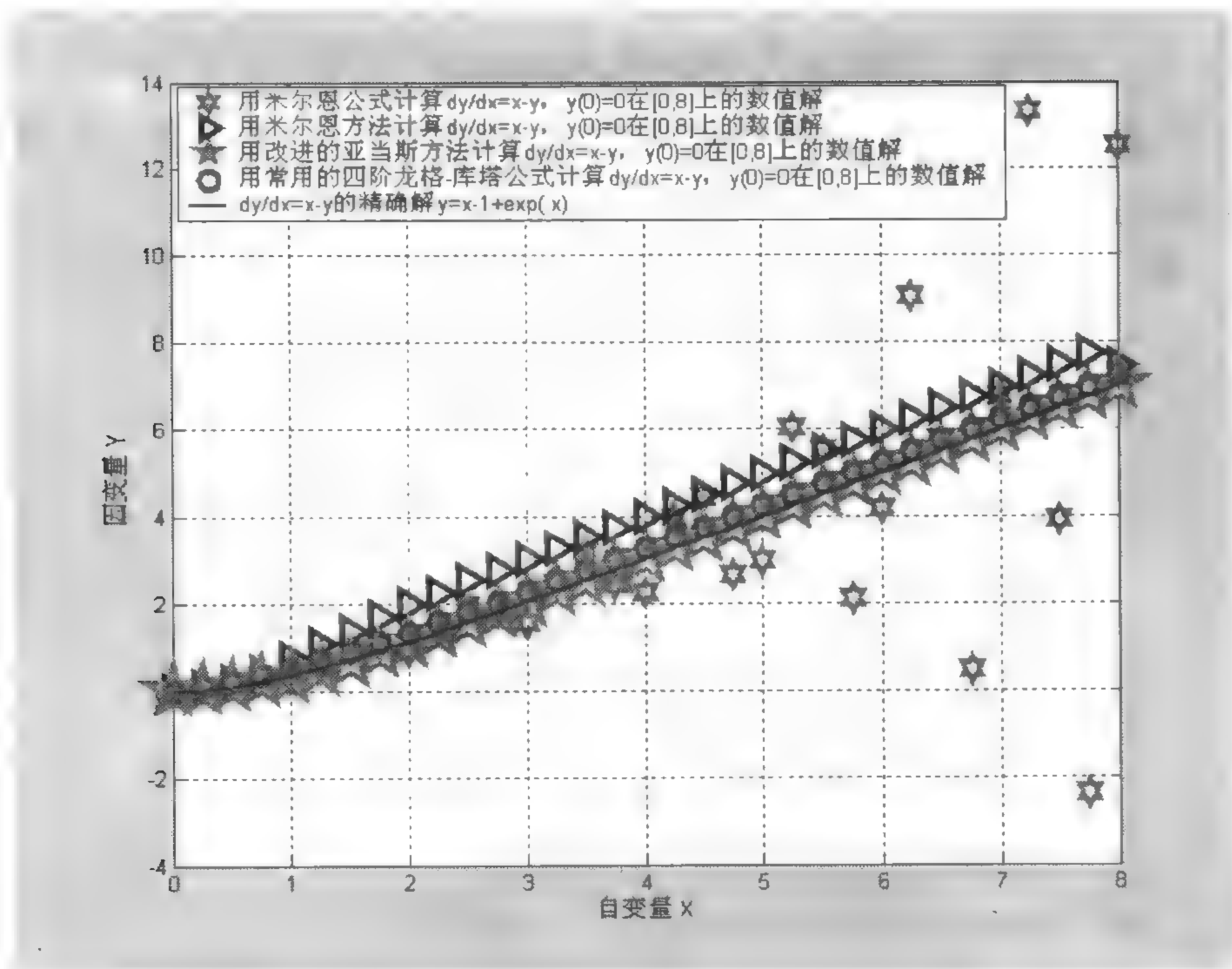


图 10-23 米尔恩方法、米尔恩公式、改进的亚当斯方法和四阶龙格-库塔公式求解

10.6.5 汉明(Hamming)公式及其 MATLAB 程序

取 $k=3,\alpha_1=\beta_2=0$,由方程组(10.48)和(10.47),可得**汉明公式**为

$$y_{n+1}=\frac{1}{8}(9y_n-y_{n-2})+\frac{3}{8}h(f_{n+1}+2f_n-f_{n-1}), \tag{10.56}$$

局部截断误差为

$$R_{n+1}=-\frac{1}{40}h^5y_n^{(5)}+O(h^6). \tag{10.57}$$

汉明公式是四阶三步隐式公式.

根据汉明公式(10.56)式编写的求解常微分方程初值问题(10.5)的MATLAB程序如下.

汉明公式求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序

输入量:*funfcn* 是函数 $f(x,y)$,x_0和 y_0是初值 $y(x_0)=y_0$,b 是自变量 x 的最大值,h 是步长.

输出量:向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值,向量 $\boldsymbol{Y}$ 的元素是利用(10.56)式求出的初值问题(10.5)在向量 $\boldsymbol{X}$ 的元素处的数值解,n 是自变量 x 取值的序号,wucha(k +1) =norm(Y(k +1) - Y(k)),并画出数值解和精确解的图形.

```
function [k,X,Y,wucha,P] = Hamming(funfcn,x0,b,y0,h)
x = x0;y = y0;p = 128; n = fix((b - x0)/h);
if n < 5,return,end;
X = zeros(p,1); Y = zeros(p,length(y)); f = zeros(p,1);
k = 1; X(k) = x; Y(k,:) = y';
for k = 2:4
x = x + h; c1 = 1/6;c2 = 1/3;c3 = 1/3;c4 = 1/6;a2 = 1/2; a3 = 1/2;
a4 = 1;b21 = 1/2;b31 = 0;
b32 = 1/2; b41 = 0;b42 = 0;b43 = 1;x1 = x + a2 * h;
x2 = x + a3 * h; x3 = x + a4 * h;
k1 = feval(funfcn,x,y); y1 = y + b21 * h * k1;k2 = feval(funfcn,x1,y1);
y2 = y + b31 * h * k1 + b32 * h * k2;k3 = feval(funfcn,x2,y2);
y3 = y + b41 * h * k1 + b42 * h * k2 + b43 * h * k3;k4 = feval(funfcn,x3,y3);
y = y + h * (c1 * k1 + c2 * k2 + c3 * k3 + c4 * k4);X(k) = x;Y(k,:) = y;k = k + 1;
end
X;Y; f(1:4) = feval(funfcn,X(1:4),Y(1:4));
for k = 4:n
f(k) = feval(funfcn,X(k),Y(k));
X(k + 1) = X(1) + h * k;
Y(k + 1) = (1/8) * (9 * Y(k) - Y(k - 2)) + (3 * h/8) * ((f(k - 2:k))' *
[ -1 2 1]');
f(k + 1) = feval(funfcn,X(k + 1),Y(k + 1)); f(k) = f(k + 1); k = k + 1;
end
for k = 2:n + 1
wucha(k) = norm(Y(k) - Y(k - 1));
end
X = X(1:n + 1);Y = Y(1:n + 1,:);n = 1:n + 1,
wucha = wucha(1:n,:);P = [n',X,Y,wucha'];
```

例 10.6.6 先利用常用四阶龙格 - 库塔公式求初值问题$\frac{dy}{dx}=1-\frac{2xy}{1+x^2}$,

$y|_{x=0}=0, 0\leqslant x\leqslant 20$ 的几个点的数值解，再利用汉明公式、米尔恩公式和改进的四阶亚当斯隐式公式及常用的四阶龙格-库塔公式求解其余的数值解，$h=\frac{1}{2}$，并计算它与精确解 $y=\dfrac{x+\frac{1}{3}x^3}{1+x^2}$ 的误差，在同一图形窗口画出精确解和数值解的图形.

解 (1) 对此问题的汉明公式的具体形式为

$$\begin{cases} f_n=1-\dfrac{2x_ny_n}{1+x_n^2}, 0\leqslant x_n\leqslant 20, n=4,5,6,\cdots, \\ y_{n+1}=\dfrac{1}{8}(9y_n-y_{n-2})+\dfrac{3h}{8}(f_{n+1}+2f_n-f_{n-1}). \end{cases}$$

(2) 编写并保存名为 funfcn.m 和 fun.m 的 M 文件如下

```
function f = funfcn(x,y)
f =1 -(2.*x.*y)./(1 +x.^2);
function y = fun (x)
y =(x +1/3.*x.^3)./(1 +x.^2);
```

(3) 在 MATLAB 工作窗口输入下面的程序

```
>> x0 =0;b =20;y0 =0;h =1/2;
[k,X,Yh,wucha,P] = Hamming(@funfcn,x0,b,y0,h)
y =(X +1/3 *X.^3)./(1 +X.^2);
[k,X,Y,wucha,P] = Milne(@funfcn,x0,b,y0,h)
[k,X,Y1,wucha,P] =Adams4y1(@funfcn,x0,b,y0,h), c1 =1/6;c2 =1/3;
b31 =0; b41 =0;b42 =0;b43 =1; c3 =1/3;c4 =1/6;a2 =1/2; a3 =1/2;
a4 =1;b21 =1/2; b32 =1/2;
C =[c1,c2,c3, c4,a2, a3, a4,b21,b31,b32,b41,b42,b43];
[k,X,Y2,fxy,wch,wucha,P] =RK4(@funfcn,@fun,x0,b,C,y0,h)
plot(X,Yh,'bh',X,Y,'m*',X,Y1,'gp',X,Y2,'ro',X,y,'k-'),
grid,xlabel('自变量 X'),ylabel('因变量 Y')
legend('用汉明公式计算 y =1 -2xy/(1 +x^2),y(0) =0 在[0,20]上的数值解','用米尔恩公式计算 y =1 -2xy/(1 +x^2),y(0) =0 在[0,20]上的数值解','用改进的四阶亚当斯方法计算 y =1 -2xy/(1 +x^2),y(0) =0 在[0,20]上的数值解','用常用的四阶龙格-库塔公式计算 y =1 -2xy/(1 +x^2),y(0) =0 在[0,20]上的数值解','y =1 -2xy/(1 +x^2)的精确解 y =(x +1/3x^3)/(1 +x^2)')
wchY =abs(y -Y),wchYh =abs(y -Yh),wchY1 =abs(y -Y1),
wchY2 =abs(y -Y2), m =zeros(1,k), for n =1:k,m(1,n) =n -1,end,
[m',X,y,Yh,Y,Y1,Y2,wchYh,wchY,wchY1,wchY2],
```

运行后屏幕显示图 10-24(a)和计算结果(见表 10-18).

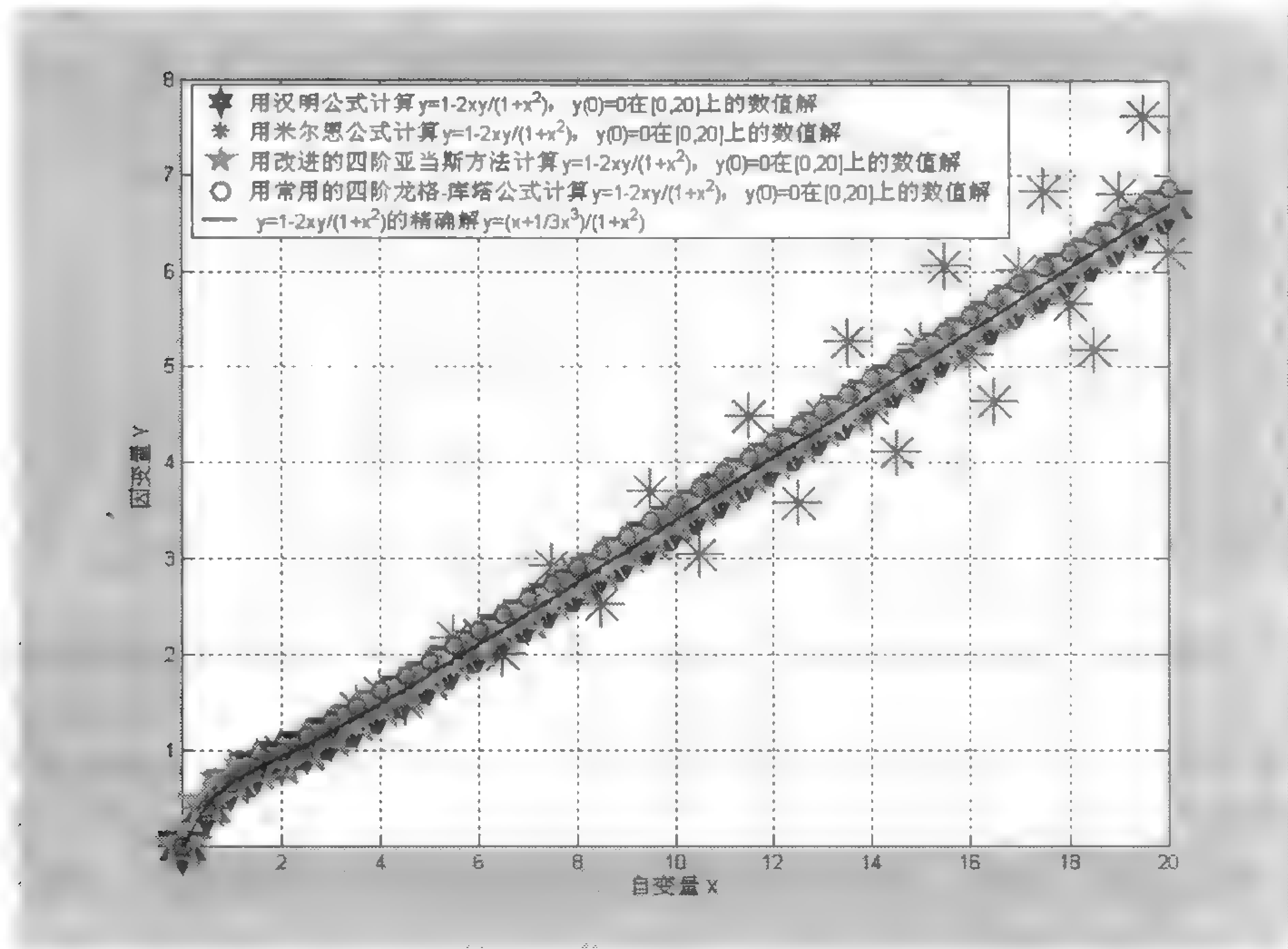


图 10－24(a) 取 $h=\frac{1}{2}$ 时，汉明公式、米尔恩公式、改进的四阶亚当斯方法和常用的四阶龙格－库塔公式求初值问题

表 10－18 取 $h=\frac{1}{2}$ 时，用汉明公式、米尔恩公式、改进的四阶亚当斯方法和常用的四阶龙格－库塔公式计算结果

m	X	y	Yh	Y	$Y1$	$Y2$	$wchYh$	$wchY$	$wchY1$	$wchY2$
1	0.500 0	0.433 3	0.395 5	0.395 5	0.433 2	0.395 5	0.037 8	0.037 8	0.000 1	0.037 8
2	1.000 0	0.666 7	0.640 7	0.640 7	0.666 3	0.640 7	0.025 9	0.025 9	0.000 4	0.025 9
3	1.500 0	0.807 7	0.824 8	0.824 8	0.807 4	0.824 8	0.017 1	0.017 1	0.000 3	0.017 1
4	2.000 0	0.933 3	0.929 7	0.990 1	0.855 2	0.988 4	0.003 6	0.056 8	0.078 2	0.055 0
5	2.500 0	1.063 2	1.042 6	1.013 2	1.006 5	1.145 7	0.020 6	0.050 1	0.056 7	0.082 5
6	3.000 0	1.200 0	1.179 9	1.118 8	1.159 7	1.301 8	0.020 1	0.081 2	0.040 3	0.101 8
7	3.500 0	1.342 8	1.322 7	1.445 6	1.311 8	1.458 3	0.020 0	0.102 9	0.031 0	0.115 5
8	4.000 0	1.490 2	1.472 4	1.586 0	1.465 2	1.615 6	0.017 8	0.095 8	0.025 0	0.125 4
9	4.500 0	1.641 2	1.625 2	1.497 3	1.620 4	1.774 0	0.015 9	0.143 9	0.020 8	0.132 8
10	5.000 0	1.794 9	1.780 8	1.700 9	1.777 2	1.933 3	0.014 1	0.094 0	0.017 7	0.138 4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
31	15.500 0	5.209 5	5.206 7	6.055 6	5.206 4	5.372 7	0.002 8	0.846 1	0.003 1	0.163 2

续表

m	X	y	Yh	Y	$Y1$	$Y2$	$wchYh$	$wchY$	$wchY1$	$wchY2$
32	16.000 0	5.374 8	5.372 2	5.125 4	5.371 9	5.538 3	0.002 7	0.249 5	0.002 9	0.163 4
33	16.500 0	5.540 3	5.537 7	4.637 5	5.537 5	5.703 9	0.002 5	0.902 8	0.002 8	0.163 6
34	17.000 0	5.705 7	5.703 3	6.016 4	5.703 1	5.869 5	0.002 4	0.310 7	0.002 7	0.163 8
35	17.500 0	5.871 3	5.869 0	6.838 6	5.868 8	6.035 3	0.002 3	0.967 3	0.002 5	0.163 9
36	18.000 0	6.036 9	6.034 7	5.665 5	6.034 5	6.201 0	0.002 2	0.371 4	0.002 4	0.164 1
37	18.500 0	6.202 6	6.200 5	5.180 4	6.200 3	6.366 8	0.002 1	1.022 2	0.002 3	0.164 2
38	19.000 0	6.368 3	6.366 3	6.807 4	6.366 1	6.532 7	0.002 0	0.439 1	0.002 2	0.164 4
39	19.500 0	6.534 1	6.532 2	7.617 6	6.532 0	6.698 6	0.001 9	1.083 5	0.002 1	0.164 5
40	20.000 0	6.699 9	6.698 1	6.193 2	6.697 9	6.864 5	0.001 9	0.506 7	0.002 0	0.164 6

从图 10－24(a)和表 10－18 可见,汉明公式解初值问题得到的数值解序列$\{Y_h\}$不但收敛,而且与精确解的绝对误差误差最小;其次是改进的四阶亚当斯方法计算的绝对误差;用四阶龙格－库塔公式计算的数值解的绝对误差误差排居第三;米尔恩方法计算的误差最大,且得到的数值解序列$\{Y\}$不稳定. 但是,步长 h 越小,这四种方法计算的数值解的绝对误差误差越小,米尔恩方法计算的数值解序列$\{Y\}$趋于稳定(参见取 $b=20, h=\frac{1}{20}$时,用这四种方法计算的图 10－24(b)). 请读者分别取不同的 b 和 h 的值计算,观察其误差的变化.

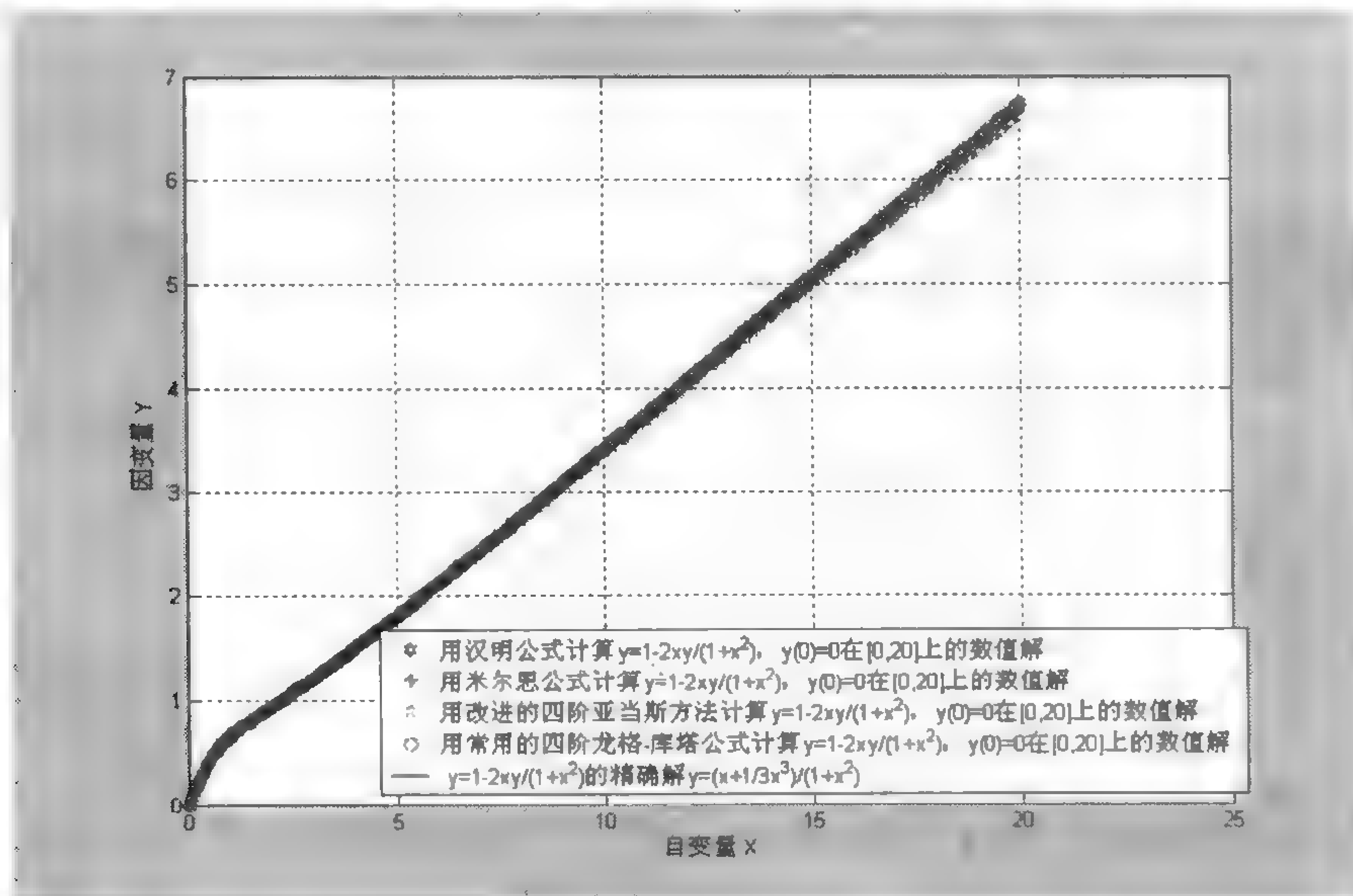


图 10－24(b) 取 $h=\frac{1}{20}$时,用四种方法求初值问题

10.6.6 预测 – 校正系统及其 MATLAB 程序

用显式公式计算预测值，然后用隐式公式进行校正，得到近似值 y_{n+1}，这样一组计算公式称为**预测 – 校正系统**. 一般地，**预测 – 校正系统采用同阶的隐式公式与显式公式**. 下面首先分别介绍的单环节的预测 – 校正系统有亚当斯预测 – 校正公式和米尔恩 – 汉明预测 – 校正公式，多环节的预测 – 校正系统有亚当斯预测 – 校正公式和米尔恩 – 汉明预测 – 校正公式，然后用 MATLAB 程序实现

（一）单环节的亚当斯预测 – 校正公式及其 MATLAB 程序

首先用四阶亚当斯显式公式(10.50)作为预测，然后用四阶亚当斯隐式公式(10.52)作校正，构成**单环节的亚当斯预测 – 校正公式**

$$\begin{cases} p_{n+1}=y_n+\dfrac{h}{24}(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}), \\ y_{n+1}=y_n+\dfrac{h}{24}[9f_{n+1}(x_{n+1},p_{n+1})+19f_n-5f_{n-1}+f_{n-2}]. \end{cases} \tag{10.58}$$

因为(10.58)式是四阶公式，所以它的起步值除了 y_0 给定的以外，y_1,y_2,y_3 通常用四阶龙格 – 库塔公式计算. 有时为提高精度，校正公式可迭代进行多次，但是迭代次数一般不超过 3 次.

根据亚当斯预测 – 校正公式(10.58)编写的求解常微分方程初值问题(10.5)的 MATLAB 程序如下.

用单环节的亚当斯预测 – 校正公式求解常微分方程初值问题(10.5)的数值解的 MATLAB 主程序

输入量：*funfcn* 是函数 $f(x,y)$，x_0 和 y_0 是初值 $y(x_0)=y_0$，b 是自变量 x 的最大值，h 是步长.

输出量：向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值，向量 $\boldsymbol{Y}$ 的元素是利用(10.58)式求出初值问题(10.5)在向量 $\boldsymbol{X}$ 的元素处的数值解，n 是自变量 x 取值的序号，wucha(k+1) = norm(Y(k+1) - Y(k))，并画出数值解和精确解的图形.

```
function [k,X,Y,wucha,P] = dAdamsyx(funfcn,x0,b,y0,h)
x = x0;y = y0;p = 128; n = fix((b - x0)/h);
if n < 5,return,end;
X = zeros(p,1); Y = zeros(p,length(y));
f = zeros(p,1);k = 1; X(k) = x; Y(k,:) = y';
for k = 2:4
c1 = 1/6;c2 = 1/3;c3 = 1/3;c4 = 1/6;a2 = 1/2; a3 = 1/2;
a4 = 1;b21 = 1/2;b31 = 0;b32 = 1/2; b41 = 0;b42 = 0;b43 = 1;
```

```
x1 =x + a2 * h; x2 =x + a3 * h; x3 =x + a4 * h; k1 = feval(funfcn,x,y);
y1 =y + b21 * h * k1; x =x + h; k2 = feval(funfcn,x1,y1);
y2 =y + b31 * h * k1 + b32 * h * k2; k3 = feval(funfcn,x2,y2);
y3 =y + b41 * h * k1 + b42 * h * k2 + b43 * h * k3; k4 = feval(funfcn,x3,y3);
y =y + h * (c1 * k1 + c2 * k2 + c3 * k3 + c4 * k4); X(k) =x; Y(k,:) =y;
end
X;Y;f = feval(funfcn,X(1:4),Y(1:4));f = f',
for k =4:n
X(k +1) =X(1) + h * k;f(k) = feval(funfcn,X(k),Y(k));
P =Y(k) +(h/24) * ((f(k -3:k)) * [ -9 37 -59 55]');
f =[f(2) f(3) f(4) feval(funfcn,X(k +1),P)],
Y(k +1) =Y(k) +(h/24) * (f * [1 -5 19 9]');
f(4) = feval(funfcn,X(k +1),Y(k +1));k =k +1;
end
for k =1:n
    wucha(k +1) =norm(Y(k +1) -Y(k));
end
X -X(1:n +1);Y -Y(1:n +1,:);n =1:n +1,
wucha =wucha(1:n,:);P =[n',X,Y,wucha'];
```

例 10.6.7 先利用常用的四阶龙格－库塔公式求例 10.6.1 的初值问题，算出几个点的数值解，再分别利用单环节的亚当斯预测－校正公式、四阶亚当斯显式公式(10.50)、四阶亚当斯隐式公式(10.52)求解常微分方程初值问题，$h=\frac{1}{5}$，并计算它与精确解 $y=\dfrac{x+\frac{1}{3}x^3}{1+x^2}$ 的误差，在同一图形窗口画出精确解和数值解的图形.

解 (1) 对此问题的单环节的亚当斯预测－校正公式的具体形式为

$$\begin{cases} f_n = 1-\dfrac{2x_n y_n}{1+x_n^2}, 0\leqslant x_n\leqslant 2, n=4,5,6,\cdots, \\ p_{n+1}=y_n+\dfrac{h}{24}(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}), \\ y_{n+1}=y_n+\dfrac{h}{24}[9f_{n+1}(x_{n+1},p_{n+1})+19f_n-5f_{n-1}+f_{n-2}]. \end{cases}$$

(2) 在 MATLAB 工作窗口输入下面的程序

```
>> x0 =0;b =2;y0 =0;h =1/5;
subplot(3,1,1)
[k,X,Y1,wucha1,P1] =dAdamsyx(@ funfcn,x0,b,y0,h)
y -(X +1/3 * X.^3)./(1 +X.^2);plot(X,Y1,'mh',X,y,'bo'),grid
```

```
legend('用单环节的亚当斯预测 - 校正公式计算的数值解','精确解 y = (x + 1/3x^3)/(1 + x^2)')
wch1 = abs(y -Y1),[P1,y,wch1]
title('dy/dx = 1 - (2xy)/(1 + x^2),y(0) = 0 在[0,2]上的数值解和精确解的图形')
subplot(3,1,2)
[k,X,Y2,wucha2,P2] = Adams4x(@funfcn,x0,b,y0,h)
y = (X + 1/3 * X.^3)./(1 + X.^2);plot(X,Y2,'mh',X,y,'bo'),grid
legend('用四阶亚当斯显式公式计算的数值解','精确解 y - (x + 1/3x^3)/(1 + x^2)')
wch2 = abs(y - Y2),[P2,y,wch2]
subplot(3,1,3)
[k,X,Y,wucha,P] = Adams4y1(@funfcn,x0,b,y0,h),y = (X + 1/3 * X.^3)./(1 + X.^2);
plot(X,Y,'mh',X,y,'bo'),grid,xlabel('自变量 X'),ylabel('因变量 Y')
legend('用四阶亚当斯隐式公式计算的数值解','精确解 y = (x + 1/3x^3)/(1 + x^2)')
wch = abs(y - Y),[P,y,wch],A = [X,y,Y1,Y2,Y,wch1,wch2,wch]
```

运行后屏幕显示图 10－25 和计算结果(略). 从图 10－25 和计算结果可见单环节的亚当斯预测－校正公式的数值解与精确解的绝对误差比四阶亚当斯显式公式和四阶亚当斯隐式公式都小.

(二) 多环节的亚当斯预测－校正公式

为减少一次迭代所产生的误差,常常用局部截断误差进一步修正预测值与校正值得到更精确的预测－校正公式. 下面对亚当斯预测－校正公式(10.58)进行讨论.

用四阶亚当斯显式公式(10.50)的局部截断误差公式(10.51)

$$y(x_{n+1}) - p_{n+1} = \frac{251}{720}h^5 y_n^{(5)} + O(h^6) \qquad (10.59)$$

减去四阶亚当斯隐式公式(10.52)的局部截断误差公式(10.53)

$$y(x_{n+1}) - y_{n+1} = -\frac{19}{720}h^5 y_n^{(5)} + O(h^6), \qquad (10.60)$$

得

$$h^5 y_n^{(5)} \approx \frac{720}{270}(y_{n+1} - p_{n+1}). \qquad (10.61)$$

将(10.61)式分别代入(10.59)式和(10.60)式,得

$$y(x_{n+1}) - p_{n+1} \approx \frac{251}{270}(y_{n+1} - p_{n+1}), \qquad (10.62)$$

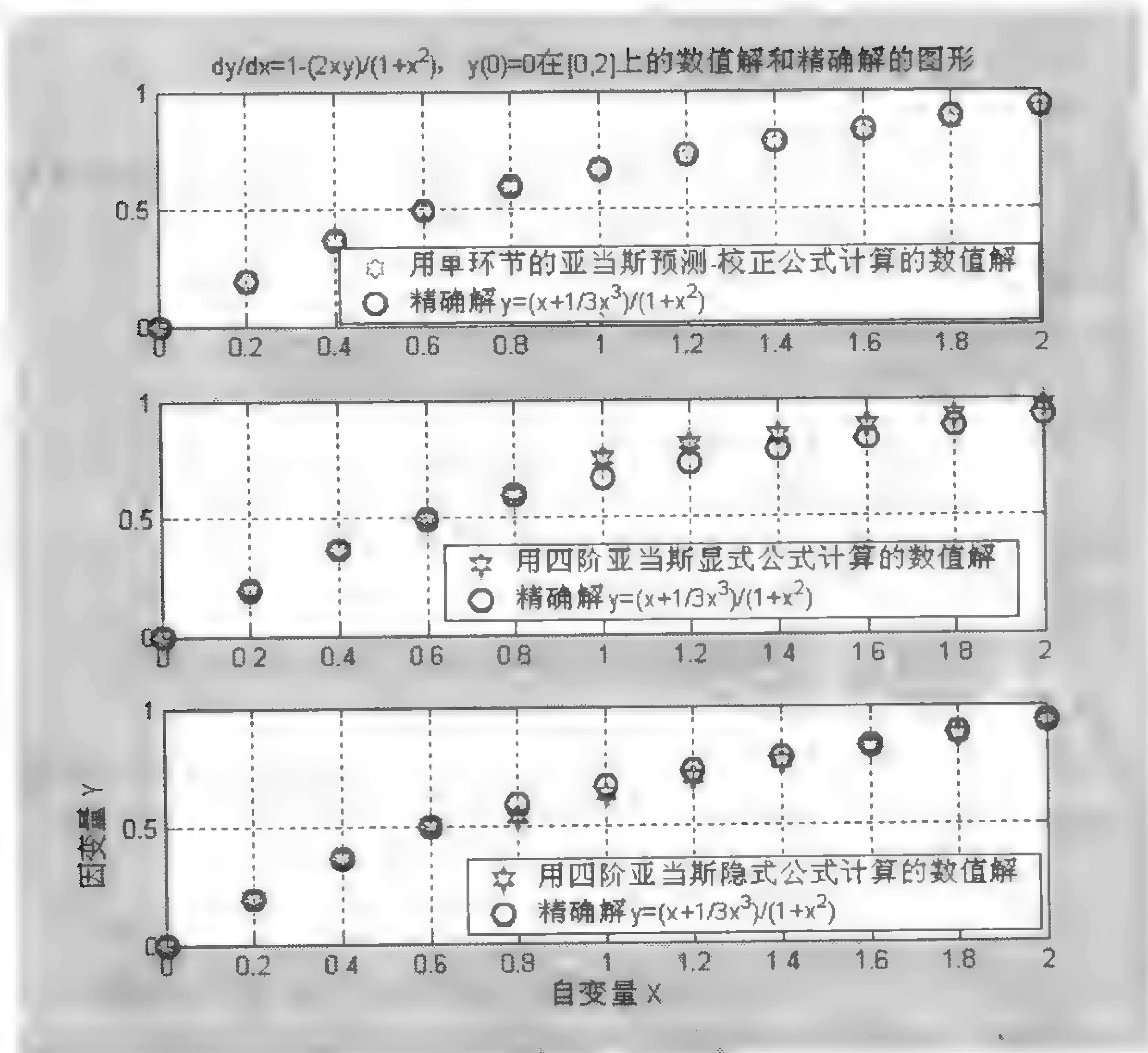


图 10-25 单环节的亚当斯预测-校正公式、四阶亚当斯显式公式和四阶亚当斯隐式公式

$$y(x_{n+1})-y_{n+1}\approx -\frac{19}{270}(y_{n+1}-p_{n+1}). \tag{10.63}$$

用(10.62)式和(10.63)式修正公式(10.58),就得到**多环节的亚当斯预测-校正公式**

$$\begin{cases} p_{n+1}=y_n+\dfrac{h}{24}(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}), & \text{预测} \\ m_{n+1}=p_{n+1}+\dfrac{251}{270}(c_n-p_n), & \text{改进} \\ c_{n+1}=y_n+\dfrac{h}{24}[9f_{n+1}(x_{n+1},m_{n+1})+19f_n-5f_{n-1}+f_{n-2}], & \text{校正} \\ y_{n+1}=c_{n+1}-\dfrac{19}{270}(c_{n+1}-p_{n+1}). & \text{改进} \end{cases} \tag{10.64}$$

上述预测-校正公式的优点是每算一步只需计算两个函数值,计算量小于四阶龙格-库塔方法,而且在计算过程中已经大致估计出误差.它们的不足之处在于必须借助于别的方法计算开始几个函数值,计算过程中不易变步长.

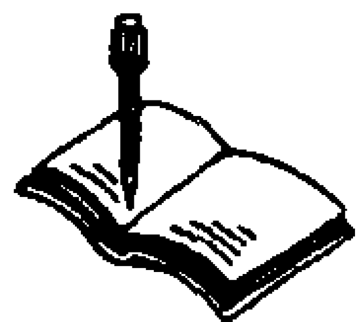
(三) 米尔恩－汉明预测－校正公式

单环节的米尔恩－汉明预测－校正公式为

$$\begin{cases} p_{n+1}=y_{n-3}+\dfrac{4h}{3}(2f_n-f_{n-1}+2f_{n-2}), \\ y_{n+1}=\dfrac{1}{8}(9y_n-y_{n-2})+\dfrac{3h}{8}[f_{n+1}(x_{n+1},p_{n+1})+2f_n-f_{n-1}]. \end{cases} \tag{10.65}$$

用推导多环节的亚当斯预测－校正公式的方法，可得**多环节的米尔恩－汉明预测－校正公式为**

$$\begin{cases} p_{n+1}=y_{n-3}+\dfrac{4h}{3}(2f_n-f_{n-1}+2f_{n-2}), \\ m_{n+1}=p_{n+1}+\dfrac{112}{121}(c_n-p_n), \\ c_{n+1}=\dfrac{1}{8}(9y_n-y_{n-2})+\dfrac{3h}{8}[f_{n+1}(x_{n+1},m_{n+1})+2f_n-f_{n-1}], \\ y_{n+1}=c_{n+1}+\dfrac{9}{121}(c_{n+1}-p_{n+1}). \end{cases} \tag{10.66}$$



习题 10.6

1. 分别用四阶亚当斯显式公式(10.50)和四阶亚当斯隐式公式(10.52)求下列初值问题的数值解，$h=\dfrac{1}{15}$并计算与精确解的误差，画出精确解和数值解的图形.

(1) $\begin{cases} \dfrac{dy}{dx}=y-\dfrac{2x}{y}, 0\leqslant x\leqslant 2, \\ y|_{x=0}=1; \end{cases}$　　(2) $\begin{cases} \dfrac{dy}{dx}=x+y, 0\leqslant x\leqslant 1, \\ y|_{x=0}=1. \end{cases}$

2. 分别利用单环节和多环节的亚当斯预测－校正公式求第 1 题中初值问题的数值解，$h=\dfrac{1}{15}$，并计算与精确解的误差，画出精确解和数值解的图形.

3. 先分别用四阶亚当斯显式公式(10.50)和四阶亚当斯隐式公式(10.52)求下列初值问题的数值解，然后分别利用单环节和多环节的米尔恩－汉明预测－校正公式计算，$h=\dfrac{1}{16}$，并计算与精确解的误差，画出精确解和数值解的图形.

(1) $\begin{cases} \dfrac{dy}{dx}=\dfrac{3y}{1+x}, 0\leqslant x\leqslant 1, \\ y|_{x=0}=1; \end{cases}$　　(2) $\begin{cases} \dfrac{dy}{dx}=x^2+x^3y, 0\leqslant x\leqslant 2, \\ y|_{x=1}=1. \end{cases}$

4. 分别利用单环节和多环节的亚当斯预测－校正公式计算积分 $I(x)=\int_0^x e^{t^2}dt$ 在 $x=1$ 附近的近似值，取 $h=0.2$，计算结果保留四位小数.

5. 用汉明公式和米尔恩公式求下列初值问题的数值解，$h=\frac{1}{16}$，并计算与精确解的误差，画出精确解和数值解的图形.

(1) $\begin{cases}(1+x^2)\dfrac{dy}{dx}+2xy=4x^2, 1\leqslant x\leqslant 2,\\ y|_{x=0}=1;\end{cases}$ (2) $\begin{cases}\dfrac{dy}{dx}=x^2+x^3y, 0\leqslant x\leqslant 2,\\ y|_{x=1}=1.\end{cases}$

6. 分别利用单环节的米尔恩－汉明预测－校正公式、多环节的米尔恩－汉明预测－校正公式和多环节的亚当斯预测－校正公式求初值问题$\frac{dy}{dx}=x-y, 0\leqslant x\leqslant 8, y|_{x=0}=0$ 的数值解，$h=\frac{1}{4}$，并计算它们与精确解的误差，在同一图形窗口画出精确解和数值解的图形.

10.7 一阶(高)阶微分方程(组)的数值解及其 MATLAB 程序

10.7.1 一阶微分方程组的数值解及其 MATLAB 程序

(一) 一阶微分方程组的数值解

设有一阶常微分方程组的初值问题

$$\begin{cases}\dfrac{dy_k}{dx}=f_k(x,y_1(x),y_2(x),\cdots,y_m(x)),\\ y_k(x_0)=y_{k0}\end{cases}(k=1,2,\cdots,m). \tag{10.67}$$

如果 $\boldsymbol{y}=(y_1,y_2,\cdots,y_m)^{\mathrm T}$，$\boldsymbol{y}_0=(y_{10},y_{20},\cdots,y_{m0})^{\mathrm T}$，$\boldsymbol{f}=(f_1,f_2,\cdots,f_m)^{\mathrm T}$，则初值问题(10.67)可以写成如下向量形式

$$\begin{cases}\dfrac{d\boldsymbol{y}}{dx}=\boldsymbol{f}(x,\boldsymbol{y}),\\ \boldsymbol{y}(x_0)=\boldsymbol{y}_0,\end{cases} \tag{10.68}$$

如果向量函数 $\boldsymbol{f}(x,\boldsymbol{y})$ 在区域 $D:x\in[x_0,b],\boldsymbol{y}\in\mathbf{R}^n$ 上对 x 连续，且对 $\boldsymbol{y}$ 满足利普希茨(Lipschitz)条件，即存在常数 L，使得

$$\|\boldsymbol{f}(x,\boldsymbol{y}_1)-\boldsymbol{f}(x,\boldsymbol{y}_2)\|\leqslant L\|\boldsymbol{y}_1-\boldsymbol{y}_2\|$$

对所有 $x\in[x_0,b]$，$\boldsymbol{y}_1,\boldsymbol{y}_2\in\mathbf{R}^n$ 都成立，则初值问题(10.68)式在闭区间$[a,b]$上有唯一解 $\boldsymbol{y}=\boldsymbol{y}(x)$.

初值问题(10.68)与(10.5)形式上完全相同，故对初值问题(10.5)的所有的各种数值解法可全部用于求解初值问题(10.68)，只需将 y 换成向量 $\boldsymbol{y}$，$f(x,y)$ 换成向量函数 $\boldsymbol{f}(x,\boldsymbol{y})$ 即可.

例如，初值问题(10.68)式的**向前欧拉公式**

$$\begin{cases} \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(x_n, \boldsymbol{y}_n), n = 0, 1, \cdots, \\ x_n = x_0 + nh, \end{cases} \tag{10.69}$$

其中 $\boldsymbol{y}_n = (y_{n1}, y_{n2}, \cdots, y_{nm})^{\mathrm{T}}$,其分量形式为

$$y_{nk+1} = y_{kn} + hf_k(x, y_{n1}, y_{n2}, \cdots, y_{nm})^{\mathrm{T}} (k = 1, 2, \cdots, m).$$

改进的欧拉公式可写作

$$\begin{cases} \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \dfrac{h}{2}(\boldsymbol{k}_1 + \boldsymbol{k}_2), \\ \boldsymbol{k}_1 = \boldsymbol{f}(x_n, \boldsymbol{y}_n), \\ \boldsymbol{k}_2 = \boldsymbol{f}(x_{n+1}, \boldsymbol{y}_n + h\boldsymbol{k}_1). \end{cases} \tag{10.70}$$

常用的四阶龙格 - 库塔公式为

$$\begin{cases} \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \dfrac{h}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4), \\ \boldsymbol{k}_1 = \boldsymbol{f}(x_n, \boldsymbol{y}_n), \\ \boldsymbol{k}_2 = \boldsymbol{f}\left(x_n + \dfrac{h}{2}, \boldsymbol{y}_n + \dfrac{h\boldsymbol{k}_1}{2}\right), \\ \boldsymbol{k}_3 = \boldsymbol{f}\left(x_n + \dfrac{h}{2}, \boldsymbol{y}_n + \dfrac{h\boldsymbol{k}_2}{2}\right), \\ \boldsymbol{k}_4 = \boldsymbol{f}(x_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_3), \end{cases} \tag{10.71}$$

其中 $\boldsymbol{k}_j = (k_{j1}, k_{j2}, \cdots, k_{jm})^{\mathrm{T}} (j = 1, 2, 3, 4)$,其分量形式为

$$\begin{cases} y_{nj+1} = y_{nj} + \dfrac{h}{6}(k_{1j} + 2k_{2j} + 2k_{3j} + k_{4j}), \\ k_{1j} = f_j(x_n, y_{n1}, y_{n2}, \cdots, y_{nm}), \\ k_{2j} = f_j\left(x_n + \dfrac{h}{2}, y_{n1} + \dfrac{hk_{11}}{2}, \cdots, y_{nm} + \dfrac{hk_{1m}}{2}\right), \\ k_{3j} = f_j\left(x_n + \dfrac{h}{2}, y_{n1} + \dfrac{hk_{21}}{2}, \cdots, y_{nm} + \dfrac{hk_{2m}}{2}\right), \\ k_{4j} = f_j(x_n + h, y_{n1} + hk_{31}, \cdots, y_{nm} + hk_{3m}) \end{cases} (j = 1, 2, \cdots, m). \tag{10.72}$$

一般地,初值问题(10.68)的**单步法的公式为**

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{g}(x_n, \boldsymbol{y}_n, h), \tag{10.73}$$

其中 $\boldsymbol{g} = (g_1, g_2, \cdots, g_m)^{\mathrm{T}}$是 $m+2$ 元的 m 维向量函数. 例如,当 $m = 2$ 时,初值问题(10.68)的具体形式为

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}x} = f_1(x, y_1, y_2), y_1(x_0) = y_{10}, \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}x} = f_2(x, y_1, y_2), y_2(x_0) = y_{20}, \end{cases} \tag{10.74}$$

(10.74)式的常用的四阶龙格－库塔公式分量形式为

$$\begin{cases} y_{n1+1}=y_{n1}+\frac{h}{6}(k_{11}+2k_{21}+2k_{31}+k_{41}), \\ y_{n2+1}=y_{n2}+\frac{h}{6}(k_{12}+2k_{22}+2k_{32}+k_{42}), \\ k_{11}=f_1(x_n,y_{n1},y_{n2},\cdots,y_{nm}), \\ k_{12}=f_2(x_n,y_{n1},y_{n2},\cdots,y_{nm}), \\ k_{21}=f_1\left(x_n+\frac{h}{2},y_{n1}+\frac{hk_{11}}{2},\cdots,y_{nm}+\frac{hk_{1m}}{2}\right), \\ k_{22}=f_2\left(x_n+\frac{h}{2},y_{n1}+\frac{hk_{11}}{2},\cdots,y_{nm}+\frac{hk_{1m}}{2}\right), \\ k_{31}=f_1\left(x_n+\frac{h}{2},y_{n1}+\frac{hk_{21}}{2},\cdots,y_{nm}+\frac{hk_{2m}}{2}\right), \\ k_{32}=f_2\left(x_n+\frac{h}{2},y_{n1}+\frac{hk_{21}}{2},\cdots,y_{nm}+\frac{hk_{2m}}{2}\right), \\ k_{41}=f_1(x_n+h,y_{n1}+hk_{31},\cdots,y_{nm}+hk_{3m}), \\ k_{42}=f_2(x_n+h,y_{n1}+hk_{31},\cdots,y_{nm}+hk_{3m}). \end{cases} \tag{10.75}$$

（二）一阶微分方程组的数值解的 MATLAB 程序

用 MATLAB 计算一阶微分方程组的数值解的常用方法有两类：一类是根据求微分方程组的数值解的公式编写 MATLAB 程序计算；另一类是根据表10－12中解常微分方程初值问题的 MATLAB 库函数计算.

1. 利用 MATLAB 库函数计算一阶微分方程组的数值解

根据表 10－12 中解常微分方程初值问题的 MATLAB 库函数计算一阶微分方程组的数值解，首先需要根据一阶常微分方程组

$$\frac{dy_1}{dx}=y_2,\frac{dy_2}{dx}=y_3,\cdots,\frac{dy_m}{dx}=f(x,y_1,y_2,\cdots,y_m),x\in[a,b]$$

建立相关名为 dzdx.m 的 M 函数文件如下

```
function  dy = dzdx(x,y)
dy(1) =y(2);
dy(2) = y(3);
……;
dy(n) = f(x,dy(1),dy(2),……,dy(m));
dy =[dy(1);dy(2);……;dy(m)];
```

然后，根据表 10－12 中解常微分方程初值问题的 MATLAB 库函数调用 dzdx.m

计算,其调用格式为

```
[x,y]=odeij('dzdx',H,y0)
```

输入的量:H 是方程组中自变量的取值范围 $H=[a,b]$;$\boldsymbol{y}_0$ 是初始值 $y_k(x_0)=y_{k0}(k=1,2,\cdots,m)$ 构成的向量 $\boldsymbol{y}_0=[y_1(x_0),y_2(x_0),\cdots,y_m(x_0)]$.

输出的量:$\boldsymbol{x}$ 是方程组中自变量取值构成的向量;$\boldsymbol{y}$ 是方程组的解对应于 $\boldsymbol{x}$ 的函数值构成的矩阵.

例 10.7.1 求微分方程组 $\begin{cases} z'_1=z_2, \\ z'_2=5(1-z_1^2)z_2-z_1 \end{cases}$ 在区间 $H=[0,60]$ 上满足条件:$x=0$ 时,$z_1=1,z_2=2$ 的特解.

解 (1) 建立名为 dzdx1.m 的 M 函数文件

```
function  dz=dzdx1(x,z)
dz(1)=z(2);dz(2)=5*(1-z(1)^2)*z(2)-z(1);
dz=[dz(1);dz(2)];
```

(2) 调用 dzdx1.m 求解,在 MATLAB 工作窗口输入程序

```
>> H=[0,60]; z0=[1; 2];[x,z]=ode113('dzdx1',H,z0);
plot(x,z(:,1),'g-',x,z(:,2),'r:')
xlabel('轴\it x'); ylabel('轴\it y')
legend('是方程组解 z1 的曲线','是方程组解 z2 的曲线')
```

运行后求得解函数 $z_1=z_1(x)$ 和 $z_2=z_2(x)$ 的图形如图 10-26 所示.

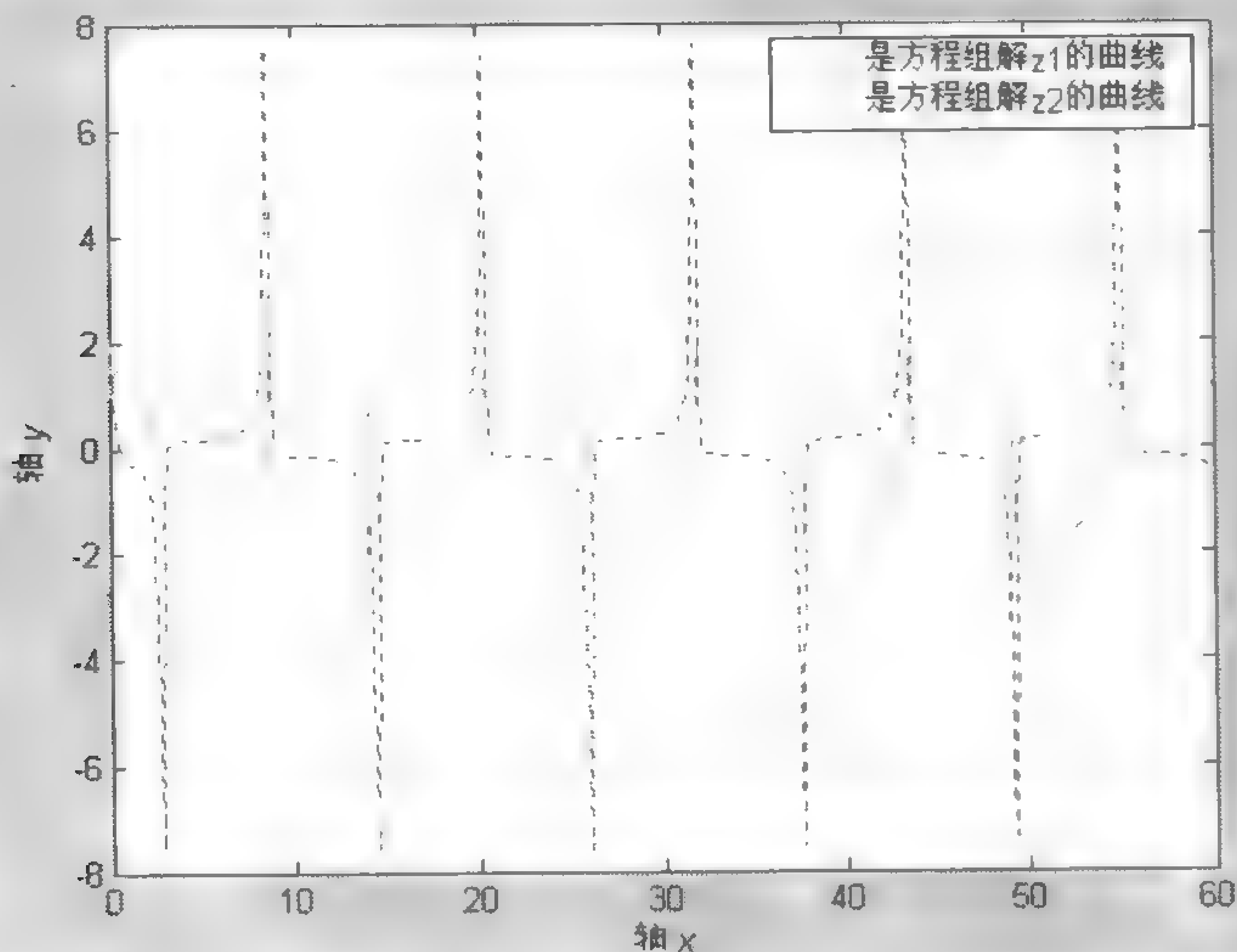


图 10-26

例 10.7.2 求微分方程组 $\begin{cases} z'_1 = z_2, \\ z'_2 = z_3, \\ z'_3 = x^{-1}z_3 - 3x^{-2}z_2 + 2x^{-3}z_1 + 9x^3\sin x \end{cases}$ 在区间 $H=[0.1,60]$ 上满足条件：$x=0.1$ 时，$z_1=1, z_2=1, z_3=1$ 的特解.

解 (1) 建立名为 dzdx2.m 的 M 函数文件

```
function  dz = dzdx2(x,z)
dz(1) = z(2);dz(2) = z(3);
dz(3) = z(3)*x^(-1)-3*x^(-2)*z(2)+2*x^(-3)*z(1)+9*x^3*
sin(x);
dz = [dz(1);dz(2);dz(3)];
```

(2) 调用 dzdx2.m 求解，在 MATLAB 工作窗口输入程序

```
>> H=[0.1,60]; z0=[1;1;1];[x,z]=ode15s('dzdx2',H,z0);
plot(x,z(:,1),'g-',x,z(:,2),'b*',x,z(:,3),'mp')
xlabel('轴\it x'); ylabel('轴\it y')
legend('是方程解 z1 的曲线','是方程解 z2 的曲线','是方程解 z3 的曲线')
```

运行后求得解函数 $z_1=z_1(x)$，$z_2=z_2(x)$ 和 $z_3=z_3(x)$ 的图形如图 10－27 所示.

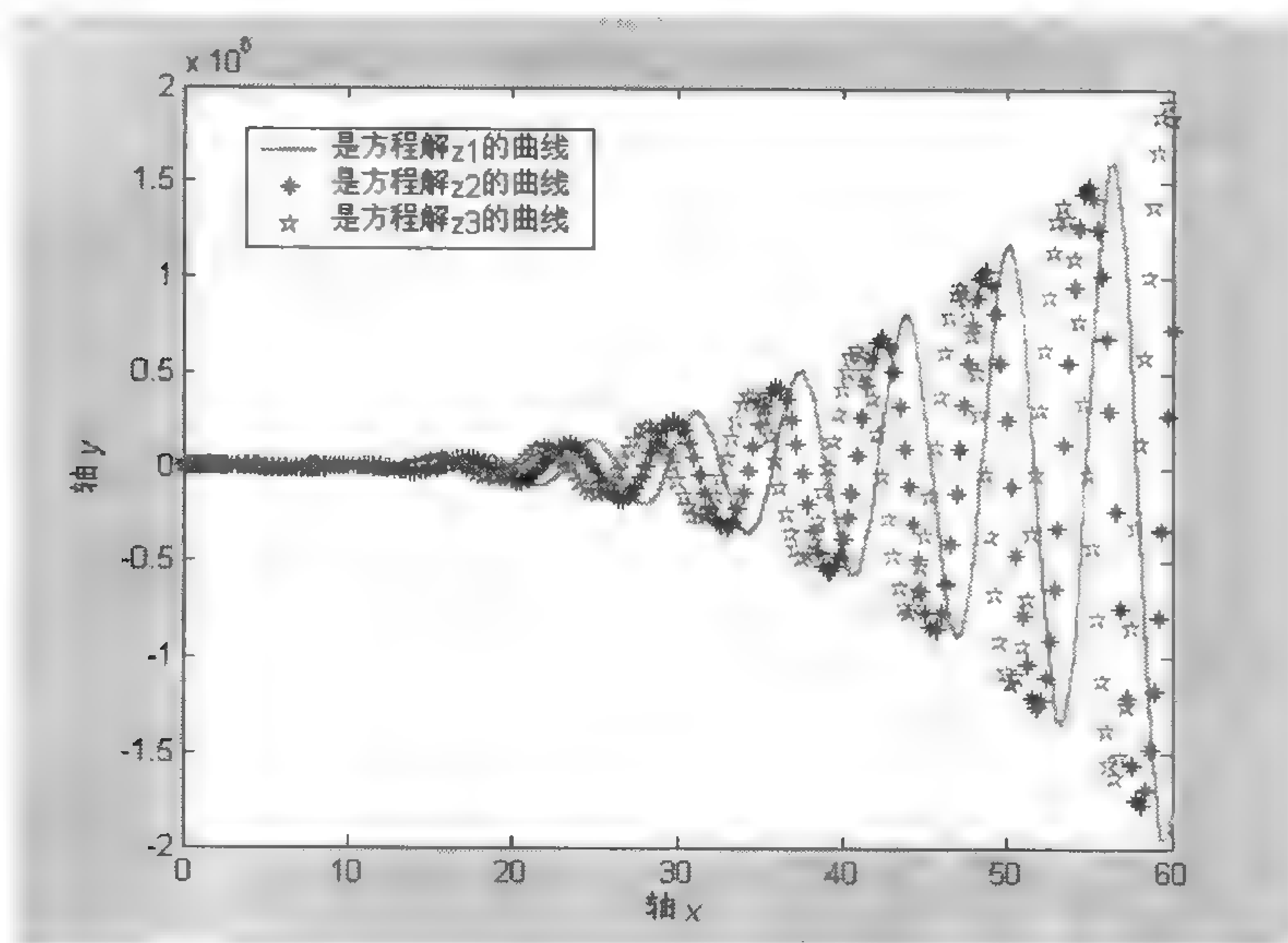


图 10－27 例 10.7.2 用 ode15s 计算

2. 根据求微分方程组的数值解的公式编写 MATLAB 程序计算

我们也可以根据一阶常微分方程组的初值问题(10.67)的具体公式编写

MATLAB 程序计算数值解.例如,根据常用的四阶龙格-库塔公式(10.75)编写求解初值问题$\frac{dy_1}{dx}=y_2,\frac{dy_2}{dx}=y_3,\cdots,\frac{dy_m}{dx}=f(x,y_1,y_2,\cdots,y_m),x\in[a,b],y_k(x_0)=y_{k0}(k=1,2,\cdots,m)$的 MATLAB 程序如下.

> **用常用的四阶龙格-库塔公式(10.75)求解常微分方程组的初值问题(10.67)的数值解的 MATLAB 主程序**
>
> **输入量**:dydx 是名为 dydx.m 相关的 M 函数文件,$\boldsymbol{CT}=[y_{10},y_{20},\cdots,y_{m0}]$是初始条件 $y_k(x_0)=y_{k0}(k=1,2,\cdots,m)$;$a$ 和 b 是自变量 x 所在的区间$[a,b]$的端点,h 是步长.
>
> **输出量**:向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值,向量 $\boldsymbol{Y}$ 的元素是利用公式(10.75)求出初值问题$\frac{dy_1}{dx}=y_2,\frac{dy_2}{dx}=y_3,\cdots,\frac{dy_m}{dx}=f(x,y_1,y_2,\cdots,y_m),x\in[a,b],y_k(x_0)=y_{k0}(k=1,2,\cdots,m)$在向量 $\boldsymbol{X}$ 的元素处的数值解,k 是自变量 x 取值的序号,wucha(k+1)=norm(Y(k+1)-Y(k)).

```
function [k,X,Y,wucha,P] = RK4z(dydx,a,b,CT,h)
n = fix((b - a)/h);
X = zeros(n+1,1);
Y = zeros(n+1,length(CT));
X = a:h:b;
Y(1,:) = CT';
for k = 1:n
k1 = feval(dydx,X(k),Y(k,:))
x2 = X(k) + h/2;y2 = Y(k,:)' + k1 * h/2;
k2 = feval(dydx,x2,y2);
k3 = feval(dydx,x2,Y(k,:)' + k2 * h/2);
k4 = feval(dydx, X(k) + h,Y(k,:)' + k3 * h);
Y(k+1,:) = Y(k,:) + h*(k1' + 2*k2' + 2*k3' + k4')/6;k = k+1;
end
for k = 2:n+1
wucha(k) = norm(Y(k) - Y(k-1)); k = k+1;
end
X = X(1:n+1);Y = Y(1:n+1,:);k = 1:n+1;wucha = wucha(1:k,:);
P = [k',X',Y,wucha'];
```

例 10.7.3 用常用的四阶龙格-库塔公式(10.75)求解例 10.7.2 的常微分方程组的初值问题的数值解,$h=0.25$,画出数值解的图形,并与例 10.7.2 比较.

解 (1) 建立名为 dydx.m 的 M 函数文件

```
function  dY = dydx(X,Y)
dY(1) = Y(2);
dY(2) = Y(3);
dY(3) = Y(3) * X^( -1) -3 * X^( -2) * Y(2) +2 * X^( -3) * Y(1) +9 * X^3 *
sin(X);
dY = [dY(1);dY(2);dY(3)];
```

(2) 调用 dydx.m 求解,在 MATLAB 工作窗口输入程序

```
>> CT = [1;1;1];h = 0.25;
[k,X,Y,wucha,P] = RK4z(@ dydx,0.1,60,CT,h),
plot(X,Y(:,1),'g -',X,Y(:,2),'b *',X,Y(:,3),'mp')
xlabel('轴 \it x'); ylabel('轴 \it y')
legend('是方程解 z1 的曲线','是方程解 z2 的曲线', '是方程解 z3 的曲线')
```

运行后求得解函数 $z_1 = z_1(x)$,$z_2 = z_2(x)$ 和 $z_3 = z_3(x)$ 的数值解(略)及其图形如图 10 - 28 所示. 比较图 10 - 28 与图 10 - 27 可知,本例题的数值解的图形与例 10.7.2 的类似. 但是,用自己编制的 RK4z.m 程序可以计算和绘制步长 h 取任意值时的常微分方程(组)的数值解及其图形,请读者分别取 $h = 0.5, 0.15, 0.05$,计算本题,并绘制出它们的图形.

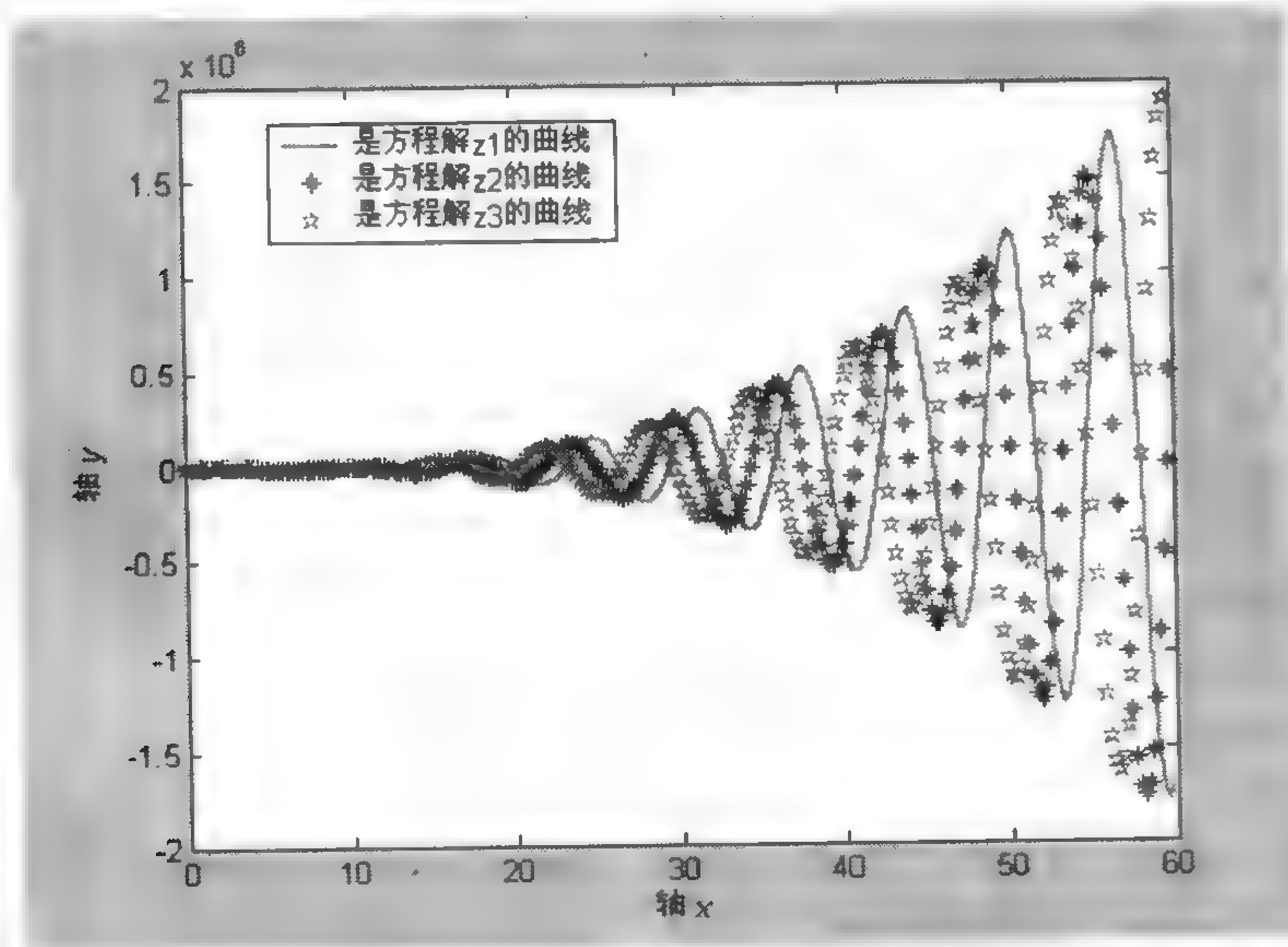


图 10 - 28 例 10.7.3 用 RK4z 计算

10.7.2 高阶微分方程(组)的数值解及其 MATLAB 程序

高阶微分方程(组)的初值问题可以通过变量代换化为一阶常微分方程组初值问题进行计算.

设有 m 阶常微分方程初值问题

$$\begin{cases}\dfrac{\mathrm{d}^m y}{\mathrm{d}x^m}=f(x,y,y',\cdots,y^{(m-1)}),x_0\leqslant x\leqslant b,\\ y(x_0)=y_0,y'(x_0)=y_{10},\cdots,y^{(m-1)}(x_0)=y_{(m-1),0}\end{cases}\quad (m=2,3,\cdots,n).\tag{10.76}$$

令 $z_1=y,z_2=y',\cdots,z_m=y^{(m-1)}$,则初值问题(10.76)就化为一阶常微分方程组

$$\begin{cases}z_1=y, & z_1(x_0)=y_0,\\ z'_1=z_2, & z_2(x_0)=y_{10},\\ \cdots\cdots & \\ z'_{m-1}=z_m, & z_{m-1}(x_0)=y_{(m-2),0},\\ z'_m=f(x,z_1,z_2,\cdots,z_m), & z_m(x_0)=y_{(m-1),0},\end{cases}\quad x_0\leqslant x\leqslant b.\tag{10.77}$$

然后用类似于例 10.7.1 中的数值方法求解问题(10.77),就可以得到(10.76)的数值解.

例 10.7.4 求微分方程 $y''-5(1-2y^4)y'+7y=0$ 在区间 $H=[0,20]$ 上满足条件:$x=0$ 时,$y=0,y'=1$ 的特解.

解 (1) 转化方程.令 $y=z_1,y'=z_2$,则原方程化为

$$\begin{cases}z'_1=z_2,\\ z'_2=5(1-2z_1^4)z_2-7z_1.\end{cases}$$

(2) 建立名为 dzdx3.m 的 M 函数文件

```
function  dz=dzdx3(x,z)
dz(1)=z(2);
dz(2)=5*(1-2*z(1)^4)* z(2)-7*z(1);dz=[dz(1);dz(2)];
```

(3) 调用 dzdx3.m 求解,在命令窗口输入

```
>> H=[0,20]; z0=[0; 1];[x,z]=ode45('dzdx3',H,z0);
plot(x,z(:,1),'g-',x,z(:,2),'r:')
xlabel('轴 \it x'); ylabel('轴 \it y')
legend('是方程解 y 的曲线','是解 y 的一阶导数')
```

运行后求得解函数 $y=z_1(t)$ 和它的导数 $y'=z_2(t)$ 的图形(图 10-29 所示.)

例 10.7.5 求解方程组 $\begin{cases}y''-5xz'=0,\\ z''+5xy'=0\end{cases}$ 在区间 $H=[-25,25]$ 上满足条件

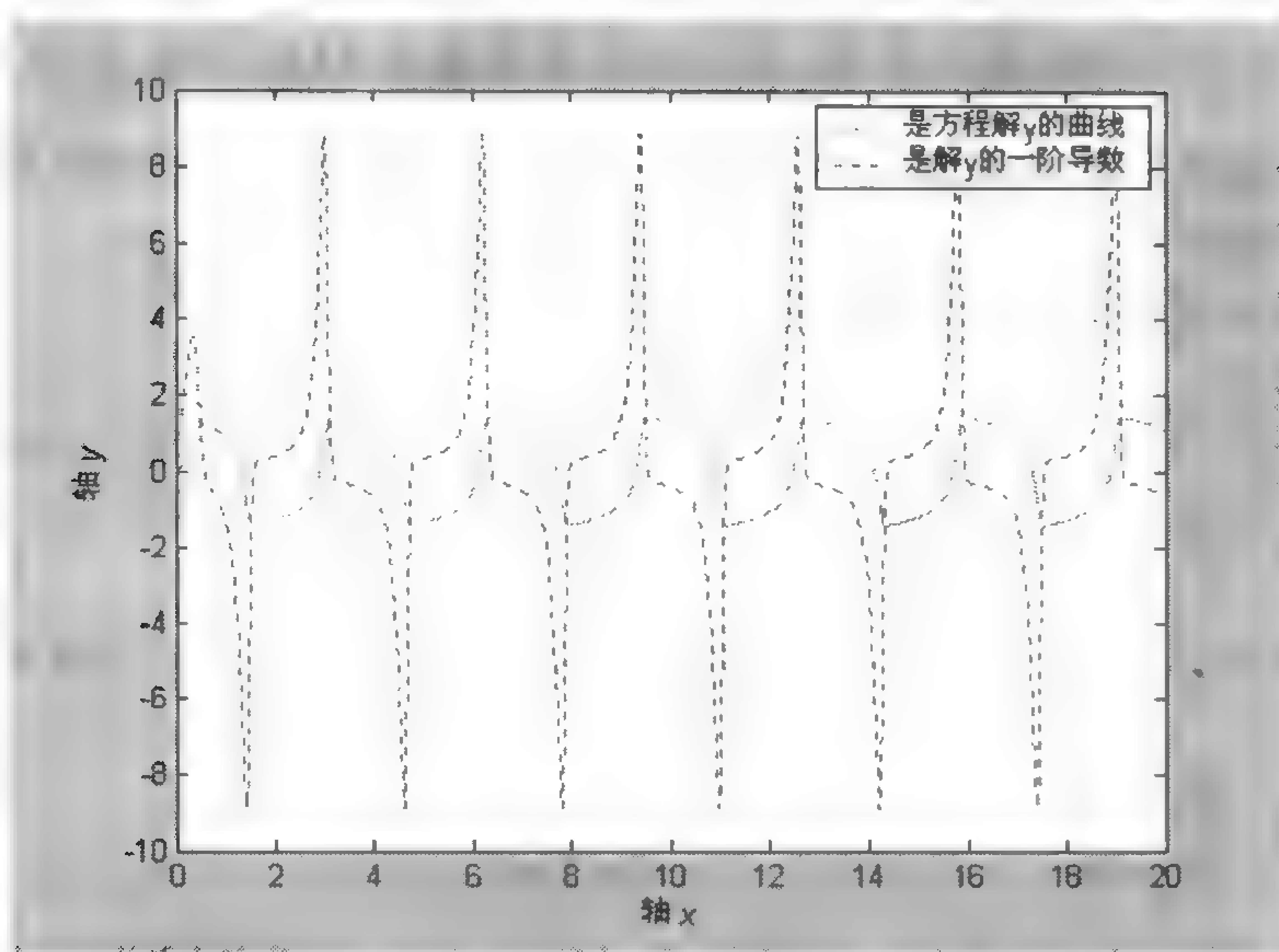


图 10－29 微分方程 $y''-5(1-2y^4)y'+7y=0$ 的解

$x=0$时$[y_0,z_0]=[0,1]$，$[y'_0,z'_0]=[-1,0]$的特解.

解 (1) 转化方程. 令 $y=z_1$，$y'=z_2$，$z=z_3$，$z'=z_4$，则原方程化为

$$\begin{cases} z'_1=z_2, \\ z'_2=5xz_4, \\ z'_3=z_4, \\ z'_4=-5xz_2. \end{cases}$$

(2) 建立名为 dzdx4.m 的 M 函数文件

```
function  dz=dzdx4(x,z)
dz(1)=z(2);dz(2)=5*x*z(4);dz(3)=z(4);dz(4)=-5*x*z(2);
dz=[dz(1);dz(2);dz(3);dz(4)];
```

(3) 调用 dzdx4.m 求解，在命令窗口输入

```
>> H=[-25,25]; z0=[0; 1; -1; 0];[t,z]=ode45('dzdx4',H,z0);
plot(z(:,1), z(:,3),'r-'),xlabel('轴\it y'); ylabel('轴\it z')
legend('是方程组的解:y与z的曲线')
```

运行后求得解函数 $y=z_1(x)$，$z=z_3(x)$的图形如图 10－30 所示.

例 10.7.6 求微分方程组

$$\begin{cases} x'=-ax+yz, \\ y'=-b(y-2z), \\ z'=cy-z-3xy \end{cases}$$

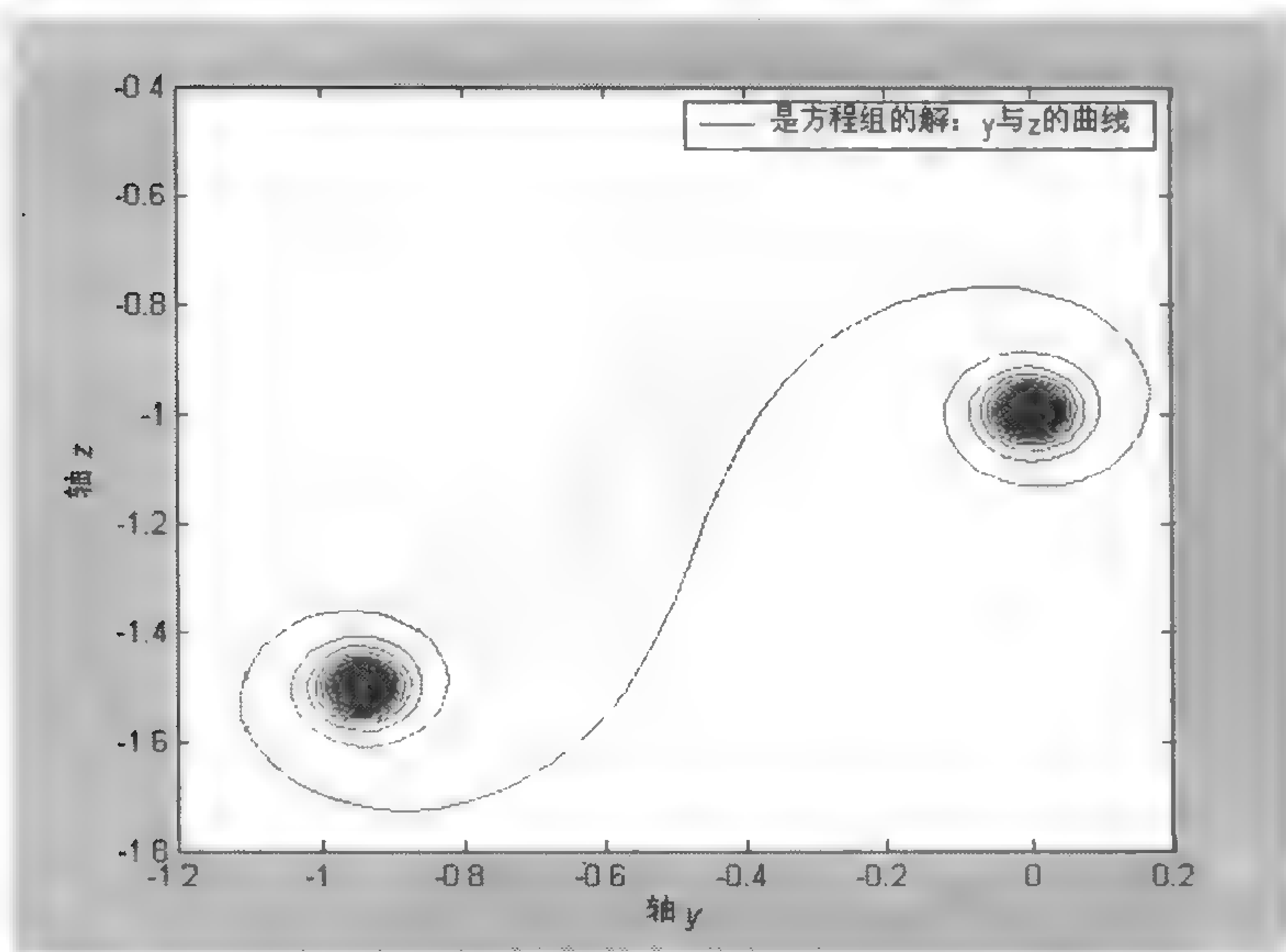


图 10-30　方程组 $y''-5xz'=0, z''+5xy'=0$ 的解

在区间$[0,75]$上满足条件:$t=0$ 时,$x=0, y=0, z=10^{-16}$ 的特解,其中 $a=3$, $b=11, c=29$

解　(1) 转化方程. 令 $x=z_1, y=z_2, z=z_3$,可得

$$\begin{cases} z'_1 = -az_1 + z_2z_3, \\ z'_2 = -b(z_2-2z_3), \\ z'_3 = cz_2 - z_3 - 3z_1z_2. \end{cases}$$

(2) 建立名为 dzdt5.m 的 M 文件

```
function  dz = dzdt5(t,z)
a = 3; b = 11; c = 29;
dz(1) = -a * z(1) + z(2) * z(3);dz(2) = -b * (z(2) -2 * z(3));
dz(3) = c * z(2) - z(3) -3 * z(2) * z(1);dz = [dz(1);dz(2);dz(3)];
```

(3) 调用 dzdt5.m 求解,在命令窗口输入

```
>> H = [0,75]; z0 = [0; 0;10^( -16)];[t,z] = ode45('dzdt5',H,z0)
plot3(z(:,1), z(:,2), z(:,3),'r-')
xlabel('轴\it x'); ylabel('轴\it y'); zlabel('轴\it z')
title('空间曲线是方程组的解:z 是 x 和 y 的函数')
```

运行后求得解函数 $x=z_1(t), y=z_2(t), z=z_3(t)$ 及其图形如图 10-31 所示(具有吸引子的空间曲线).

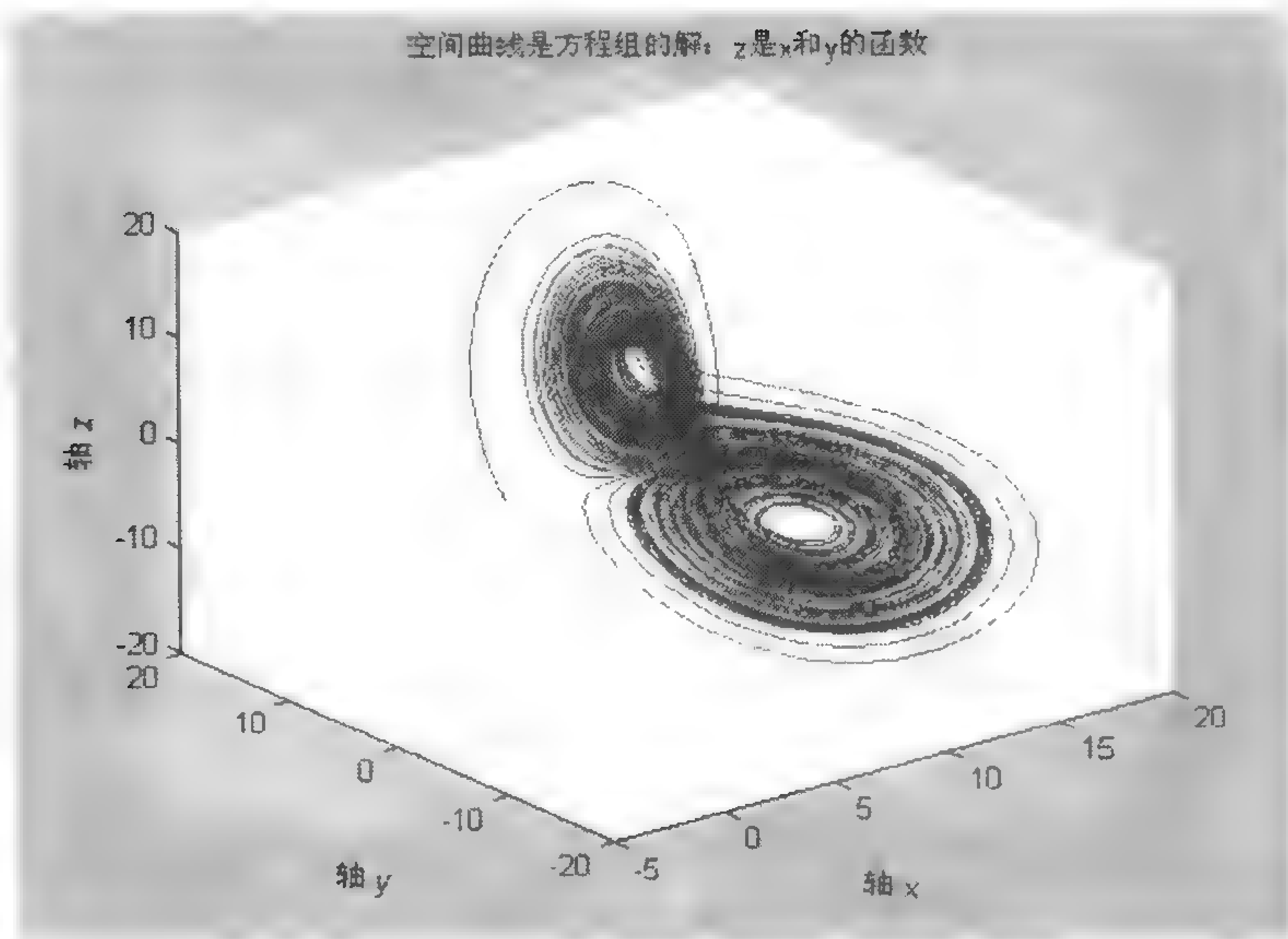


图 10-31 方程组 $x' = -ax + yz, y' = -b(y-2z), z' = cy - z - 3xy$ 的解

例 10.7.7 单摆运动求解过程

这是一个我们熟悉的物理模型,可看作工程技术中一些振动问题的简化. 图 10-32 中一根长 l 的(无弹性的)细线,一端固定,另一端悬挂一质量为 m 的小球,在重力作用下小球处于竖直的平衡位置. 使小球偏离平衡位置一个小的角度,然后让它自由,小球就会沿圆弧摆动. 在不考虑空气阻力的情况下,小球将作一定的周期简谐运动. 图 10-32 中以 $\theta=0$ 为平衡位置,以右边为正方向建立摆角 θ 的坐标系. 在小球摆动过程中的任一位置 θ,小球所受重力沿运动轨迹方向的分力为 $-mg\sin\theta$(负号表示力的方向与 θ 的正向相反),利用牛顿第二定律即得微分方程

$$ml\theta'' = -mg\sin\theta. \tag{10.78}$$

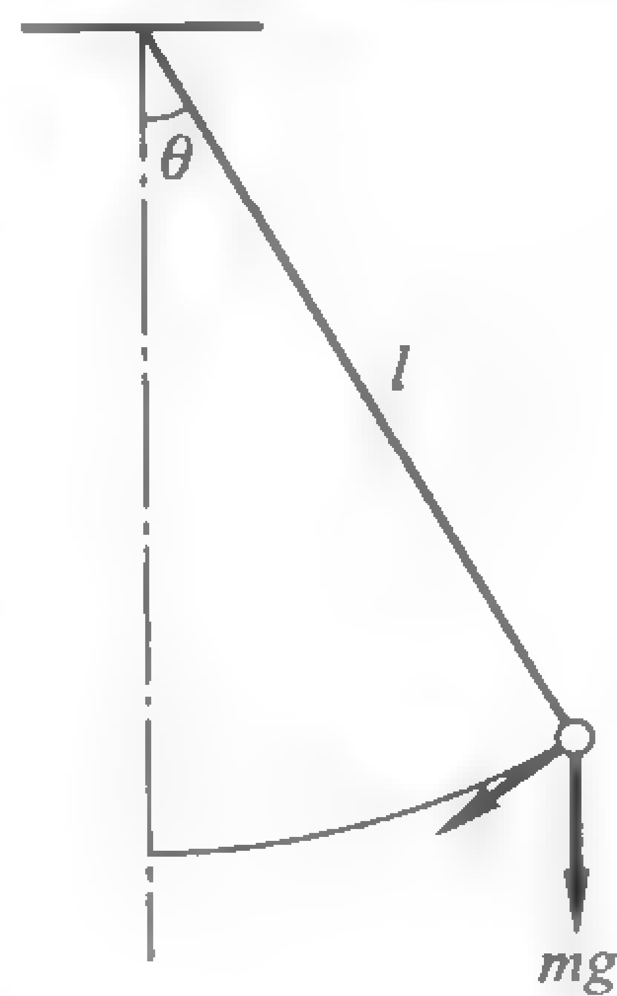


图 10-32 单摆运动

设小球初始偏离角度为 θ_0,且无初速度,则方程的初始条件为

$$\theta(0) = \theta_0, \theta'(0) = 0. \tag{10.79}$$

求解(10.78),(10.79)时,在 θ_0不大的条件下,可将方程(10.78)中的 $\sin\theta$ 近似为 θ,于是得到线性常系数微分方程

$$\theta'' + \frac{g}{l}\theta = 0. \tag{10.80}$$

容易算出方程(10.80)在初始条件(10.79)下的解为

$$\theta(t) = \theta_0 \cos \omega t, \omega = \sqrt{\frac{g}{l}}. \tag{10.81}$$

由解(10.81)可知,简谐运动的周期为 $T = 2\pi\sqrt{\frac{l}{g}}$.

现在的问题是,当 θ_0较大时,仍用 θ 近似 $\sin\theta$,误差太大了. 而方程(10.78)没有解析解. 试用数值方法在 θ_0等于 10°和 30°两种情况下求解(设 $l = 25$ cm),画出 $\theta(t)$的图形,并与近似解(10.81)的结果比较.

描述单摆运动规律的式(10.78)是 2 阶微分方程,无解析解,为了用 MATLAB求数值解,需先将它化为方程组. 令 $x_1 = \theta, x_2 = \theta'$,则方程(10.78)化为

$$x'_1 = x_2, x'_2 = -\frac{g}{l}\sin x_1. \tag{10.82}$$

初始条件(10.79)式表为

$$x_1(0) = x_{10}, x_2(0) = 0. \tag{10.83}$$

(10.82)、(10.83)式中 $g = 9.8, l = 25, x_{10}$为 10° = 0.1745 弧度及 30° = 0.5236 弧度两种情况. 对于近似解(10.79)式,周期 $T = 2\pi\sqrt{\frac{25}{9.8}} \cong 10(\text{s})$.

建立名为 danbai.m 的 M 文件

```
function xdot = danbai(t,x)
g = 9.8;l = 25;xdot(1) = x(2);
xdot(2) = -g/l * sin(x(1));xdot = [xdot(1);xdot(2)];
```

输入程序如下($x_{10} = 0.1745, x_{20} = 0.5236$)

```
H = [0,10]; a1 = 0.1745;x10 = [a1,0];[t,x1] = ode23(@danbai,H,
x10);
a2 = 0.5236;x20 = [a2,0];
[t,x2] = ode23(@danbai,H,x20);g = 9.8;l = 25;w = sqrt(g/l);
y1 = a1 * cos(w * t);y2 = a2 * cos(w * t);wu1 = y1 - x1(1:34,1);
wu2 = y2 - x2(1:34,1);
n = 1:34;[n',t,x1(1:34,1),y1,wu1,x2(:,1),y2,wu2]
plot(t,x1(1:34,1),'ro',t,y1,'g-',t,x2(:,1),'mh',t,y2,'b -.')
xlabel('轴\it t'); ylabel('轴\itθ')
legend('是方程数值解 θ 的曲线,θ0 = 10 度','是近似解 θ 的曲线, θ0 = 10 度',
'是方程数值解 θ 的曲线, θ0 = 30 度','是近似解 θ 的曲线,θ0 = 30 度')
```

运行后得到计算结果见表 10 - 19,x_1的图形如图 10 - 33.

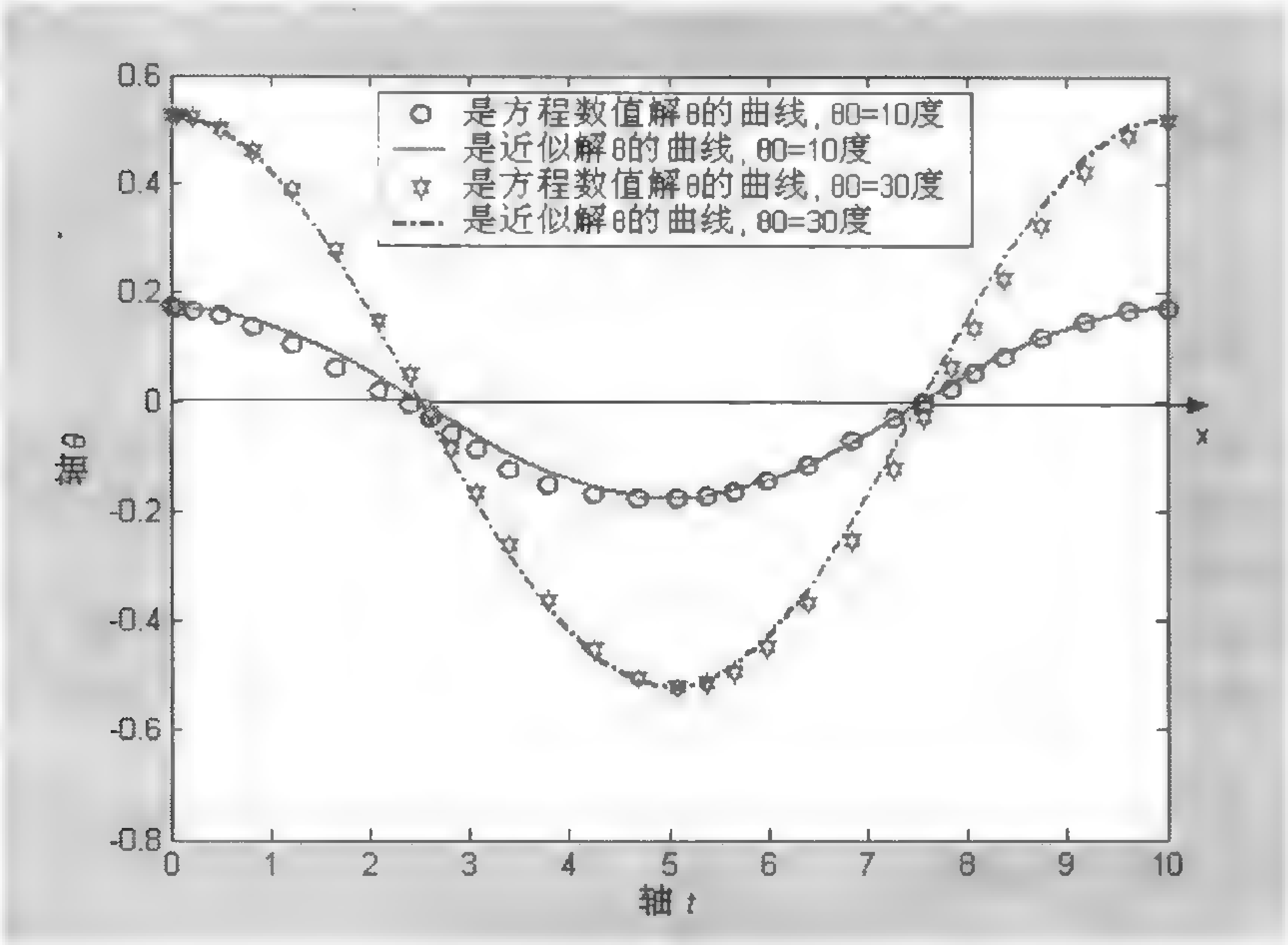


图 10－33 单摆运动初始角 $\theta_0=10°,30°$ 的近似解和数值解

表 10－19 单摆运动初始值 $\theta_0=10°,30°$ 的近似解和数值解及其绝对误差

序号 n	自变量 t	初始角 $\theta_0=10°$			初始角 $\theta_0=30°$		
		数值解 x_1	近似解 y_1	绝对误差 wu_1	数值解 x_1	近似解 y_1	绝对误差 wu_2
1	0	0.174 5	0.174 5	0	0.523 6	0.523 6	0
2	0.000 4	0.174 5	0.174 5	0.000 0	0.523 6	0.523 6	－0.000 0
3	0.002 4	0.174 5	0.174 5	0.000 0	0.523 6	0.523 6	－0.000 0
4	0.012 7	0.174 5	0.174 5	0.000 0	0.523 6	0.523 6	－0.000 0
5	0.063 7	0.173 5	0.174 4	0.000 8	0.523 2	0.523 2	－0.000 0
6	0.230 1	0.169 4	0.172 7	0.003 3	0.518 4	0.518 2	－0.000 2
7	0.485 3	0.158 9	0.166 5	0.007 6	0.500 7	0.499 6	－0.001 0
8	0.812 2	0.139 1	0.152 4	0.013 4	0.460 1	0.457 3	－0.002 8
9	1.200 3	0.106 9	0.127 5	0.020 6	0.388 0	0.382 6	－0.005 5
10	1.645 8	0.063 1	0.089 8	0.026 6	0.278 2	0.269 4	－0.008 9
11	2.096 0	0.021 0	0.044 6	0.023 6	0.145 8	0.133 8	－0.011 9
12	2.398 2	－0.003 8	0.012 1	0.015 9	0.049 8	0.036 2	－0.013 6

续表

序号 n	自变量 t	初始角 $\theta_0=10°$			初始角 $\theta_0=30°$		
		数值解 x_1	近似解 y_1	绝对误差 wu_1	数值解 x_1	近似解 y_1	绝对误差 wu_2
13	2.610 1	-0.028 6	-0.011 1	0.017 5	-0.018 7	-0.033 2	-0.014 4
14	2.822 0	-0.055 5	-0.034 0	0.021 5	-0.086 9	-0.102 0	-0.015 1
15	3.078 9	-0.087 3	-0.061 0	0.026 4	-0.167 5	-0.182 9	-0.015 5
16	3.403 2	-0.120 8	-0.092 7	0.028 1	-0.262 7	-0.278 1	-0.015 4
17	3.792 8	-0.151 1	-0.125 6	0.025 4	-0.362 8	-0.377 0	-0.014 3
18	4.256 7	-0.168 7	-0.155 1	0.013 7	-0.454 0	-0.465 3	-0.011 3
19	4.690 2	-0.173 8	-0.170 8	0.003 0	-0.506 1	-0.512 6	-0.006 5
20	5.071 5	-0.173 8	-0.174 4	-0.000 6	-0.522 7	-0.523 3	-0.000 6
21	5.377 8	-0.170 4	-0.170 1	0.000 3	-0.515 5	-0.510 3	0.005 1
22	5.647 4	-0.161 5	-0.161 1	0.000 4	-0.494 1	-0.483 4	0.010 7
23	5.986 7	-0.143 9	-0.143 4	0.000 5	-0.448 2	-0.430 2	0.018 1
24	6.385 8	-0.114 6	-0.114 3	0.000 3	-0.369 4	-0.343 0	0.026 5
25	6.842 9	-0.071 1	-0.072 4	-0.001 4	-0.252 0	-0.217 4	0.034 6
26	7.279 5	-0.028 2	-0.026 9	0.001 3	-0.120 8	-0.080 7	0.040 1
27	7.563 4	-0.001 0	0.004 0	0.005 0	-0.030 1	0.012 1	0.042 2
28	7.847 3	0.026 3	0.034 8	0.008 5	0.061 5	0.104 4	0.042 9
29	8.075 2	0.052 5	0.058 8	0.006 3	0.133 8	0.176 3	0.042 5
30	8.374 1	0.084 0	0.088 3	0.004 3	0.224 3	0.265 0	0.040 7
31	8.738 0	0.117 4	0.120 0	0.002 6	0.323 6	0.360 1	0.036 5
32	9.171 0	0.148 2	0.149 6	0.001 3	0.420 1	0.448 8	0.028 7
33	9.623 1	0.167 3	0.168 7	0.001 4	0.488 8	0.506 2	0.017 4
34	10.000 0	0.173 2	0.174 5	0.001 2	0.517 7	0.523 5	0.005 8

可以看出,初始角度为10°时近似解与数值解绝对误差不大,而初始角度为30°时,随着时间的增加二者绝对误差就很大了.



习 题 10.7

1. 求下列微分方程组的数值解,并画出图形.

(1) $\begin{cases} y' = z, \\ z' = -0.2z - \sin x \end{cases}$ 在区间 $H=[0,100]$ 上满足条件:$x=0$ 时,$y=0,z=2$;

(2) $\begin{cases} x' = -y, \\ y' = x, \\ z' = 300xy \end{cases}$ 在区间 $t\in[0,8]$ 上满足条件:$t=0$ 时,$x=3,y=4,z=12$;

(3) $\begin{cases} x'' + 4x = 0, \\ y'' + 25y = 0 \end{cases}$ 在区间 $t\in[0,2\pi]$ 上满足条件:$t=0$ 时,$x=0,y=0,x'=0,y'=40$;

(4) $\begin{cases} x' = 6\sin t, \\ y' = 6\cos t, \\ z' = 0.1t \end{cases}$ 在区间 $t\in[0,20]$ 上满足条件:$t=0$ 时,$x=0,y=0,z=0$.

2. 求下列线性微分方程的数值解,并画出图形.如果方程有初等函数形式解,请求出它的符号解.

(1) $x^2y''+xy'+y=x$ 在区间 $x\in[0,8]$ 上满足条件:$x=0$ 时,$y=1,y'=2$;

(2) $y^{(4)}+\sin x\cdot y''+ax=0$ 在区间 $x\in[0,8]$ 上满足条件:$x=0$ 时,$y=0,y'=y''=y'''=1,a=0$.

3. 求下列非线性微分方程的数值解,并画出图形.

(1) $y''-c(1-by^4)y'+ay=0$ 在区间 $H=[0,50]$ 上满足条件:$x=0$ 时,$y=2,y'=1,a=b=c=1$;

(2) $yy''+(y')^2+ax=0$ 在区间 $x\in[0,8]$ 上满足条件:$x=0$ 时,$y=1,y'=1,a=-1$.

4. 求方程组 $\begin{cases} y''-2.5xz'=0, \\ z''+2.5xy'=0 \end{cases}$ 在区间 $H=[-6.25,6.25]$ 上满足条件:$x=0$ 时 $[y_0,z_0]=[0,1]$,$[y'_0,z'_0]=[-1,0]$.

5. 求微分方程组 $x'=-ax+yz,y'=-b(y-2z),z'=cy-z-3xy$ 在区间 $[0,75]$ 上满足条件:$t=0$ 时,$x=0,y=0,z=10^{-16}$ 的特解,其中 $a=1,b=8,c=19$.

10.8 边值问题的数值解及其 MATLAB 程序

微分方程和定解条件组成定解问题通常有两种:一种是给出积分曲线在初始时刻的性态,这类条件称为初始条件,对应的定解问题称为初值问题;另一种是给出积分曲线在首末两端的性态,这类条件称为边界条件,对应的定解问题称为边值问题.例如,如果二阶微分方程

$$y''=f(x,y,y'),x\in[a,b],y\in(-\infty,+\infty), \tag{10.84}$$

边界条件为

$$y(a)=\alpha,y(b)=\beta, \tag{10.85}$$

则这类问题称为**边值问题**. 其中(10.85)式称为**第一种边界条件**.

$$y'(a)=\alpha,y'(b)=\beta$$

称为**第二种边界条件**.

$$y'(a)-\alpha_0 y(a)=\alpha_1,y'(b)+\beta_0 y(b)=\beta_1$$

称为**第三种边界条件**,其中 $\alpha_0\geqslant 0,\beta_0\geqslant 0,\alpha_0+\beta_0>0$.

比上面三种边界条件更一般的边界条件是

$$\begin{cases}c_1 y(a)+d_1 y'(a)=p_1,\\ c_2 y(b)+d_2 y'(b)=p_2,\end{cases}\tag{10.86}$$

其中 $c_i,d_i,p_i(i=1,2)$ 为常数. (10.84)和(10.86)式构成一个二阶方程的**两点边值问题**. 关于二阶常微分方程边值问题有下面的定理.

定理 10.4 设函数 $f(x,y,z)$ 在空间区域

$$\Omega=\{(x,y,z)\mid a\leqslant x\leqslant b,-\infty<y<+\infty,-\infty<z<+\infty\}$$

上连续,且对 y,z 的一阶偏导数也连续,其中 $z=y'(x)$. 如果存在常数 $L>0$,使

$$\begin{cases}f'_y(x,y,z)>0,\\ |f'_z(x,y,z)|\leqslant L\end{cases}\tag{10.87}$$

对所有 $(x,y,z)\in\Omega$ 都成立,则**边值问题**(10.84)式和(10.85)式在闭区间 $[a,b]$ 上有唯一解 $y=y(x)$.

如果(10.84)式中的右端 $f(x,y,y')=q_1(x)y'+q_2(x)y+q_3(x)$,即

$$y''=q_1(x)y'+q_2(x)y+q_3(x),x\in[a,b],\tag{10.88}$$

则称(10.88)式为**线性微分方程**. 设 $z=y'(x)$,根据定理 10.4,可以推出下面有关线性微分方程边值问题的结论.

推论 10.1 设线性微分方程(10.88)式中的右端 $f(x,y,z)=q_1(x)z+q_2(x)y+q_3(x)$ 及其偏导数 $\dfrac{\partial f}{\partial y}=q_2(x)$ 和 $\dfrac{\partial f}{\partial z}=q_1(x)$ 在空间区域

$$\Omega=\{(x,y,z)\mid a\leqslant x\leqslant b,-\infty<y<\infty,-\infty<z<\infty\}$$

上连续,其中 $z=y'(x)$. 如果存在常数 $L>0$,使

$$\begin{cases}q_2(x)>0,\\ |q_1(x)|\leqslant L=\max\limits_{a\leqslant x\leqslant b}\{|q_1(x)|\}\end{cases}\tag{10.89}$$

对所有 $(x,y,z)\in\Omega$ 都成立,则**线性边值问题**(10.88)式和(10.85)式在闭区间 $[a,b]$ 上有唯一解 $y=y(x)$.

常微分方程边值问题有很多不同的数值解法,本节只介绍两种很有用的方法:打靶法和有限差分方法.

10.8.1 打靶法及其 MATLAB 程序

打靶法的基本思想是设法将边值问题转化为初值问题来解. 下面介绍一般

的打靶法和线性打靶法.

(一) 打靶法

用打靶法求边值问题

$$\begin{cases} y''=f(x,y,y'),a\leqslant x\leqslant b, \\ y(a)=\alpha, \\ y(b)=\beta \end{cases} \tag{10.90}$$

解 $y(x)$ 的基本思想如下.

设想过点 (a,α) 的微分方程的解曲线族中,只有一条曲线过点 (b,β),它是解 $y(x)$. 初值问题

$$\begin{cases} y''=f(x,y,y'),a\leqslant x\leqslant b, \\ y(a)=\alpha, \\ y'(a)=t_k \end{cases} \tag{10.91}$$

的解曲线 $y=y(x,t_k)$ 打到点 (b,β_k),见图 10-34. 调整 t_k,使靶点接近 (b,β). 即,当 $t_k\to t$ 时,有 $\beta_k\to\beta$. 也就是说,有

$$\lim_{k\to\infty} y(b,t_k)=\beta,$$

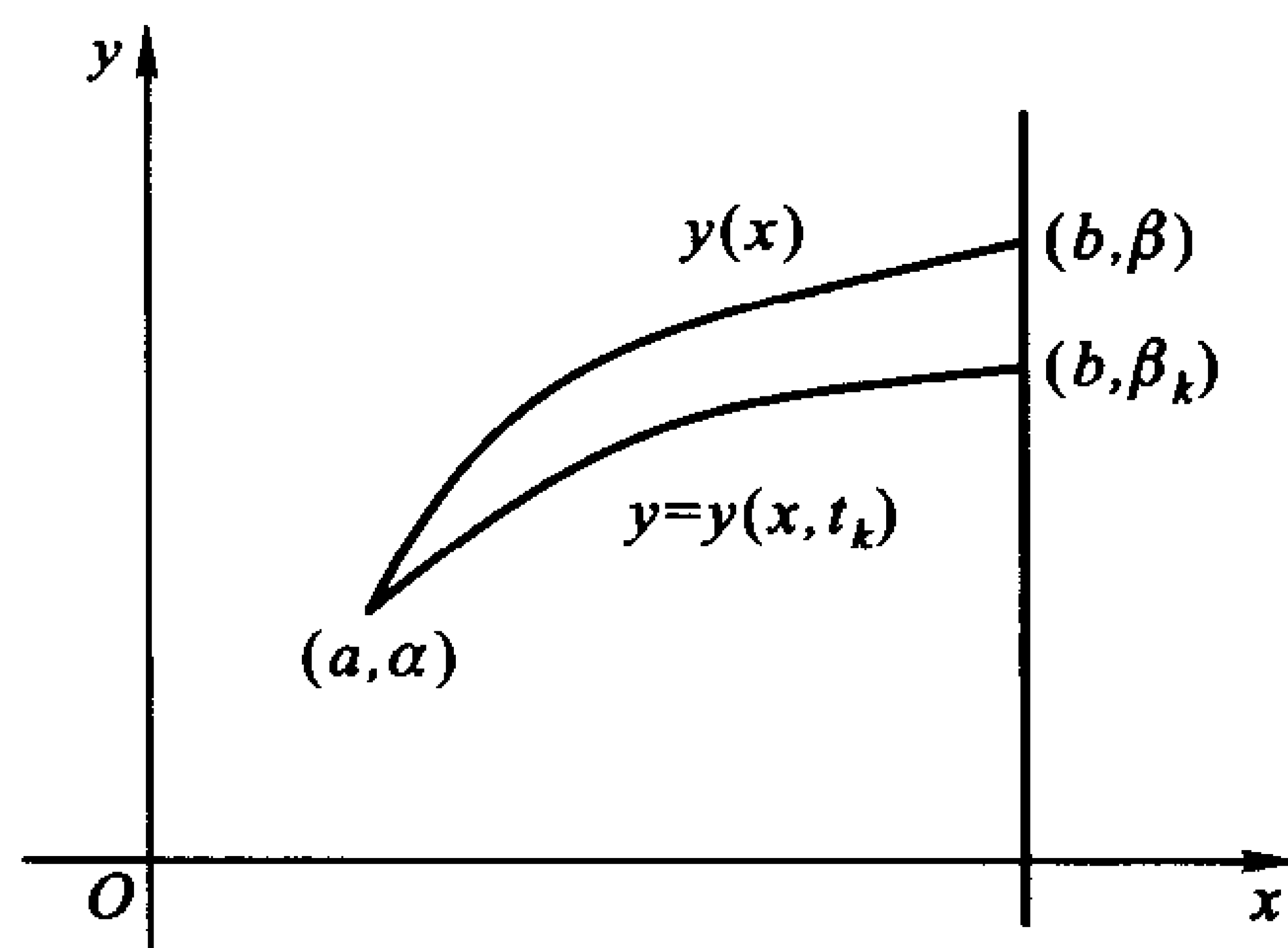


图 10-34

即,t 是方程

$$y(b,t)-\beta=0 \tag{10.92}$$

的根.

(1) 当 $y(x,t)$ 已知时,用前面介绍过的解方程的方法(如,二分法、迭代法,逐步搜索法等)求出方程(10.92)的近似根 t^*,使得

$$|y(x,t^*)-\beta|<\varepsilon,$$

则 $y(x,t^*)$ 是(10.90)的数值解,其中 ε 是预先给定的精度.

(2) 当 $y(x,t)$ 未知时,先给定一个 t_1,代入(10.91)式,得初值问题

$$\begin{cases} y' = z, \\ z' = f(x,y,z), a \leqslant x \leqslant b, \\ y(a) = \alpha, z(a) = t_1. \end{cases} \tag{10.93}$$

用例 10.7.1 中介绍的常微分方程组数值方法求初值问题(10.93)的数值解,得 $y_{1n} = \beta_1$. 如果 $|\beta_1 - \beta| < \varepsilon$,说明打靶成功,用所得到的数值解 y_{1n} 作为(10.90)的数值解. 否则,取 $t_2 = \frac{\beta}{\beta_1} t_1$,重新计算初值问题(10.91)的数值解,得 $y_{2n} = \beta_2$,再用 $|\beta_2 - \beta| < \varepsilon$ 检验打靶是否成功. 如果仍然不成功. 采用线性插值

$$t_3 = t_1 + \frac{t_2 - t_1}{\beta_2 - \beta_1}(\beta - \beta_1), \text{或 } t_3 = \frac{\beta}{\beta_2} t_2$$

再进行计算,直到达到精度 ε 为止.

说明:对于第三种边值条件,由于

$$y(a) = \frac{y'(a) - \alpha_1}{\alpha_0},$$

取 $y'(a) = t$,则有

$$\begin{cases} y'' = f(x,y,y'), a \leqslant x \leqslant b, \\ y(a) = \dfrac{t - \alpha_1}{\alpha_0}, y'(a) = t. \end{cases} \tag{10.94}$$

将(10.94)的解 $y(x,t)$ 代入第二种边界条件

$$y'(a) - \alpha_0 y(a) = \alpha_1,$$

$$y'(b) + \beta_0 y(b) = y'(b,t) + \beta_0 y(b,t) = G(t) = \beta_t$$

当 $\beta_t = \beta_1$,即是解. 这样,就同第一种边界条件一样,可以用打靶法求数值解.

(二)线性打靶法及其 MATLAB 程序

线性打靶法是求线性边值问题

$$y'' = q_1(x)y' + q_2(x)y + q_3(x), x \in [a,b], y(a) = \alpha, y(b) = \beta \tag{10.95}$$

数值解的方法. 它是利用方程的线性结构和两个特殊的初值问题辅助求(10.95)的解. 设(10.95)满足推论 1 的条件,即有唯一解,则用线性打靶法求线性边值问题(10.95)的具体步骤如下:

步骤 1 求初值问题

$$u''(x) = q_1(x)u'(x) + q_2(x)u(x) + q_3(x), x \in [a,b] \tag{10.96}$$

$u(a) = \alpha, u'(a) = 0$ 的解 $u(x)$;

步骤 2 求初值问题

$$v''(x) = q_1(x)v'(x) + q_2(x)v(x), x \in [a,b], \tag{10.97}$$

$v(a)=0, v'(a)=1$ 的解 $v(x)$，则 $y''=q_1(x)y'+q_2(x)y+q_3(x)$ 的一个解为

$$y(x)=u(x)+Cv(x). \tag{10.98}$$

步骤 3 将边值条件 $y(b)=\beta$ 分别代入(10.96)，得

$$y(b)=u(b)+Cv(b)=\beta. \tag{10.99}$$

当 $v(b)\neq 0$ 时，由(10.99)式解得 $C=\dfrac{\beta-u(b)}{v(b)}$，代入(10.98)式解得

$$y(x)=u(x)+\frac{\beta-u(b)}{v(b)}v(x). \tag{10.100}$$

(10.100)式为要求的解的形式.

根据线性打靶法和常用的四阶龙格－库塔公式(10.75)编写的求解线性边值问题(10.95)数值解的 MATLAB 主程序如下：

用线性打靶法和常用的四阶龙格－库塔公式(10.75)求解线性边值问题(10.95)数值解的 MATLAB 主程序

输入量：dydx1 和 dydx2 分别是(10.96)和(10.97)对应的 M 函数文件，alpha，beta 分别是边值条件 $y(a)=\alpha, y(b)=\beta$；a 和 b 是自变量 x 所在的区间 $[a,b]$ 的端点，n 是步数.

输出量：向量 X 的元素是自变量 x 的取值，向量 Y 的元素是用线性打靶法求出线性边值问题(10.95)在向量 X 的元素处的数值解，k 是自变量 x 取值的序号，wucha(k+1)=norm(Y(k+1)-Y(k)).

```
function [k,X,Y,wucha,P] = xxdb(dydx1,dydx2,a,b,alpha,beta,h)
n = fix((b-a)/h); X = zeros(n+1,1); CT1 = [alpha,0];
Y = zeros(n+1,length(CT1)); Y1 = zeros(n+1,length(CT1));
Y2 = zeros(n+1,length(CT1));
X = a:h:b;
Y1(1,:) = CT1;
CT2 = [0,1];Y2(1,:) = CT2;
for k = 1:n
k1 = feval(dydx1,X(k),Y1(k,:))
x2 = X(k) + h/2;y2 = Y1(k,:)' + k1 * h/2;
k2 = feval(dydx1,x2,y2);
k3 = feval(dydx1,x2,Y1(k,:)' + k2 * h/2);
k4 = feval(dydx1, X(k) + h,Y1(k,:)' + k3 * h);
Y1(k+1,:) = Y1(k,:) + h * (k1' + 2 * k2' + 2 * k3' + k4')/6,k = k+1;
end
u = Y1(:,1)
```

```
for k = 1:n
k1 = feval(dydx2,X(k),Y2(k,:))
x2 = X(k) + h/2;y2 = Y2(k,:)' + k1 * h/2;
k2 = feval(dydx2,x2,y2);
k3 = feval(dydx2,x2,Y2(k,:)' + k2 * h/2);
k4 = feval(dydx2, X(k) + h,Y2(k,:)' + k3 * h);
Y2(k + 1,:) = Y2(k,:) + h * (k1 ' + 2 * k2 ' + 2 * k3 ' + k4 ')/6,k = k + 1;
end
v = Y2(:,1)
Y = u + (beta - u(n + 1)) * v/v(n + 1)
for k = 2:n + 1
wucha(k) = norm(Y(k) - Y(k - 1)); k = k + 1;
end
X = X(1:n + 1);Y = Y(1:n + 1,:);k = 1:n + 1;wucha = wucha(1:k,:);
P = [k ',X ',Y,wucha '];
plot(X,Y(:,1),' ro ',X,Y1(:,1),' g * ',X,Y2(:,1),' mp ')
xlabel('轴 \it x '); ylabel('轴 \it y ')
legend('是边值问题的数值解 y(x)的曲线','是初值问题 1 的数值解 u(x)的曲线', '是初值问题 2 的数值解 v(x)的曲线')
title('用线性打靶法求线性边值问题的数值解的图形')
```

说明:用此程序之前需要根据(10.96)和(10.97)式,建立两个名为dydx1 .m 和 dydx2 .m 的 M 函数文件

```
function  dY1 = dydx1(X,Y)
dY1(1) = Y(2);
dY1(2) = q1(X) * Y(2) + q2(X) * Y(1) + q3(X);
dY1 = [dY1(1);dY1(2)];
function  dY2 = dydx2(X,Y)
dY2(1) = Y(2);
dY2(2) = q1(X) * Y(2) + q2(X) * Y(1);
dY2 = [dY2(1);dY2(2)];
```

例 10.8.1 用线性打靶法求线性边值问题

$$y''(x) = \frac{c_1 x}{1+x^2} y'(x) - \frac{c_2}{1+x^2} y(x) + c_3, x \in [0,6], y(0) = 1.25, y(6) = -1.25, c_1 = c_2 = 2, c_3 = 0.8$$ 的数值解,$h = 0.5$,并与精确解比较,画出它们的图形.

解 (1) 因为 $q_1(x) = \dfrac{c_1 x}{1+x^2}, q_2(x) = -\dfrac{c_2}{1+x^2}, q_3(x) = 0.8, \alpha = 1.25, \beta = -1.25$,建立两个名为 dydx1 .m 和 dydx2 .m 的 M 函数文件

```
function  dY1 = dydx1(X,Y)
c1 =2;c2 =2;c3 =0.8;
dY1(1) =Y(2);
dY1(2) =(c1 * X/(1 +X^2)) * Y(2) -(c2/(1 +X^2)) * Y(1) +c3;
dY1 =[dY1(1);dY1(2)];
function  dY2 =dydx2(X,Y)
c1 =2;c2 =2;
dY2(1) =Y(2);
dY2(2) -(c1 * X/(1 +X^2)) * Y(2) -(c2/(1 +X^2)) * Y(1);
dY2 =[dY2(1);dY2(2)];
```

(2) 求精确解. 在 MATLAB 工作窗口输入程序

```
>>y =dsolve('D2y =(2 * X/(1 +X^2)) * Dy -(2/(1 +X^2)) * y +0.8','X')
```

运行后屏幕显示结果如下

```
y =
X * C2 +(1 -X^2) * C1 -4/5 * X^2 +8/5 * X * atan(X) -2/5 * log(1 +X^
2) +2/5 * log(1 +X^2) * X^2
```

在 MATLAB 工作窗口输入程序

```
>> syms C1 C2,X =[0,6];
y = X * C2 + (1 - X.^2) * C1 - 4/5 * X.^2 +8/5 * X. * atan(X) -2/
5 * log(1 +X.^2) +2/5 * log(1 +X.^2).* X.^2
```

运行后屏幕显示结果如下

```
y =
[       C1, 6 * C2 -35 * C1 +99211677038297587/2814749767106560]
```

在 MATLAB 工作窗口输入程序

```
>>[C1,C2] =solve('C1 =1.25','6 * C2 -35 * C1 +99211677038297587/
2814749767106560 = -1.25');
syms X
y = X * C2 + (1 - X.^2) * C1 - 4/5 * X.^2 +8/5 * X. * atan(X) -2/
5 * log(1 +X.^2) +2/5 * log(1 +X.^2).* X.^2
```

运行后屏幕显示结果如下

```
y =
    1.2088219648217098859769672950885 * X +1.2500000000000000000000000000000
-2.0500000000000000000000000000000 * X^2 +8/5 * X * atan(X) -2/5 * log(1
+X^2) +2/5 * log(1 +X^2) * X^2
```

(3) 调用 xxdb.m 求此线性边值问题的数值解,在 MATLAB 工作窗口输入程序

```
>> a =0;b =6;h =0.5; alpha =1.25;beta = -1.25; c1 =2;c2 =2;c3 =0.8;
[k,X,Y,wucha,P] =xxdb(@dydx1,@dydx2,a,b,alpha,beta,h),
```

```
hold on
y=1.2088219648217098859769672950885*X+1.25-2.05*X.^2+8/5*
X.*atan(X)-2/5*log(1+X.^2)+2/5*log(1+X.^2).*X.^2;
plot(X,y,'b-'), hold off
legend('是边值问题的数值解 Y(x)的图形','是初值问题 1 的数值解 u(x)的图
形','是初值问题 2 的数值解 v(x)的图形','是边值问题的精确解 y(x)的曲线')
title('用线性打靶法求线性边值问题的数值解和精确解的图形')
n=fix((b-a)/h);
for k=1:n+1
wuchay(k)=norm(y(k)-Y(k)); k=k+1;
end
wuchay;
k=1:n+1;wuchay=wuchay(1:k,:);
P1=[k',X',y',Y, wuchay' ,wucha']
```

运行后求得此线性边值问题在自变量 X 处的数值解的值 Y,偏差 $|Y(k+1)-Y(k)|$,精确解的值 y 及其与精确解的绝对误差(见表 10-20),它们的图形(如图 10-35 所示). 从表 10-20 和图 10-35 可知,$h=0.5$ 时,本例题的数值解和精确解的误差很小.

表 10-20 例 10.8.1 的计算结果

序号 k	自变量 X	精确解 y	数值解 Y	误差 $\|y-Y\|$	偏差 $\|Y(k+1)-Y(k)\|$
1	0	1.250 0	1.250 0	0	0
2	0.500 0	1.645 9	1.639 8	0.006 1	0.389 8
3	1.000 0	1.665 5	1.654 7	0.010 7	0.014 9
4	1.500 0	1.398 8	1.384 5	0.014 2	0.270 2
5	2.000 0	0.941 8	0.925 2	0.016 6	0.459 3
6	2.500 0	0.380 8	0.362 8	0.018 0	0.562 4
7	3.000 0	-0.209 8	-0.228 2	0.018 4	0.591 0
8	3.500 0	-0.765 7	-0.783 4	0.017 7	0.555 2
9	4.000 0	-1.230 2	-1.246 3	0.016 1	0.462 9
10	4.500 0	-1.553 5	-1.567 1	0.013 6	0.320 8
11	5.000 0	-1.691 0	-1.701 0	0.010 0	0.133 9
12	5.500 0	-1.602 1	-1.607 5	0.005 5	0.093 4
13	6.000 0	-1.250 0	-1.250 0	0	0.357 5

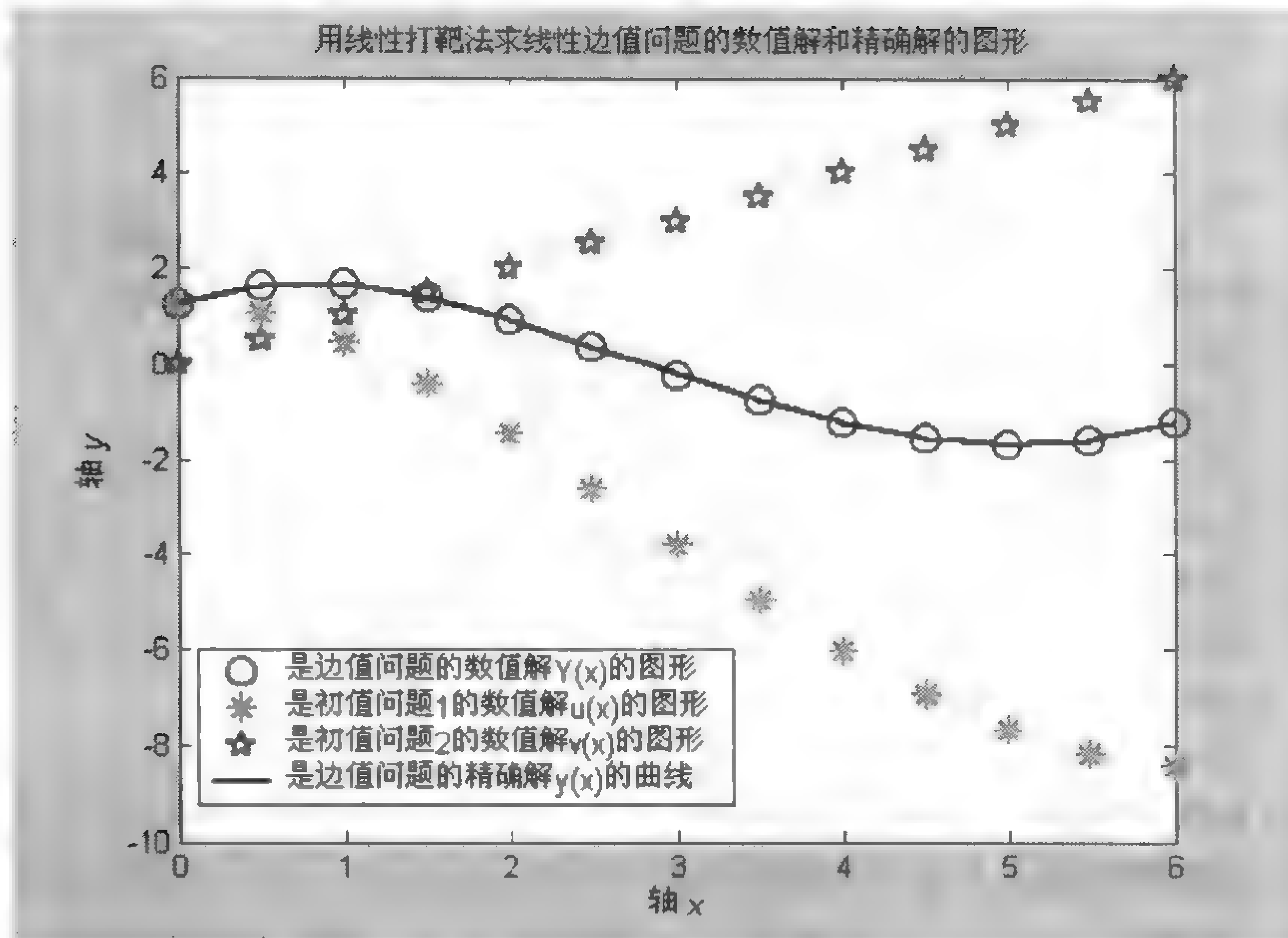


图 10－35 例 10.8.1 的数值解和精确解的图形

10.8.2 有限差分方法及其 MATLAB 程序

有限差分方法是求微分方程数值解最常用的方法. 有限差分方法的基本思想是：首先将求解区域剖分成很多有限个小区域，得到内节点的集合；在内节点上，或用差商代替微分，或用数值积分的方法将微分方程离散化，并得到截断误差，舍去截断误差，建立差分方程组；然后结合定解条件，解差分方程组，得到数值解. 有限差分方法适用于求常微分方程边值问题的数值解，用以解常微分方程初值问题的数值解时，有时有困难. 下面介绍常微分方程边值问题的有限差分方法及其 MATLAB 程序.

（一）常微分方程边值问题的有限差分方法

考虑线性边值问题

$$y''=q_1(x)y'+q_2(x)y+q_3(x),x\in[a,b],y(a)=\alpha,y(b)=\beta \quad (10.95)$$

的有限差分方法.

将求解区间 n 等分，步长 $h=\dfrac{b-a}{n}$，节点 $x_k=a+kh$ $(k=0,1,2,\cdots,n)$，其中 $x_n=b$. 在内节点 $x_k=a+kh$ $(k=1,2,\cdots,n-1)$上，用一阶中心差商

$$y'(x_k)=\frac{y(x_{k+1})-y(x_{k-1})}{2h}+O(h^2)$$

和二阶中心差商

$$y''(x_k)=\frac{y(x_{k+1})-2y(x_k)+y(x_{k-1})}{h^2}+O(h^2)$$

分别代替(10.95)式中一阶和二阶导数，且用 y_k 代替 $y(x_k)$，得到一个截断误差是 $O(h^2)$ 的差分方程

$$\frac{1}{h^2}[y_{k-1}-2y_k+y_{k+1}]+\frac{q_1(x_k)}{2h}[y_{k-1}-y_{k+1}]-q_2(x_k)y_k=q_3(x_k).$$

把(10.95)式同边值条件 $y(a)=\alpha, y(b)=\beta$ 结合起来，得到线性方程组

$$\begin{cases}-[2+h^2q_2(x_1)]y_1+\left[1-\dfrac{q_1(x_1)}{2}h\right]y_2\\ =h^2q_3(x_1)-\left[1+\dfrac{q_1(x_1)}{2}h\right]\alpha,\\ \left[1+\dfrac{q_1(x_k)}{2}h\right]y_{k-1}-[2+h^2q_2(x_k)]y_k+\left[1-\dfrac{q_1(x_k)}{2}h\right]y_{k+1}\\ =h^2q_3(x_k)(k=2,3,\cdots,n-2),\\ \left[1+\dfrac{q_1(x_{n-1})}{2}h\right]y_{n-2}-[2+h^2q_2(x_{n-1})]y_{n-1}\\ =h^2q_3(x_{n-1})-\left[1-\dfrac{q_1(x_{n-1})}{2}h\right]\beta,\end{cases}$$

即

$$\boldsymbol{AY}=\boldsymbol{B},\tag{10.101}$$

其中

$$\boldsymbol{A}=\begin{pmatrix}-(2+h^2q_2(x_1)) & 1-\dfrac{q_1(x_1)}{2}h & & & \\ 1+\dfrac{q_1(x_2)}{2}h & -(2+h^2q_2(x_2)) & 1-\dfrac{q_1(x_2)}{2}h & & \\ \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & 1+\dfrac{q_1(x_{n-2})}{2}h & -(2+h^2q_2(x_{n-2})) & 1-\dfrac{q_1(x_{n-2})}{2}h & \\ & & 1+\dfrac{q_1(x_{n-1})}{2}h & -(2+h^2q_2(x_{n-1})) & \end{pmatrix},$$

$$Y=\begin{pmatrix} y(x_1) \\ y(x_2) \\ \vdots \\ y(x_{n-2}) \\ y(x_{n-1}) \end{pmatrix},B=\begin{pmatrix} h^2q_3(x_1)-\left[1+\dfrac{q_1(x_1)}{2}h\right]\alpha \\ h^2q_3(x_2) \\ \vdots \\ h^2q_3(x_{n-2}) \\ h^2q_3(x_{n-1})-\left[1-\dfrac{q_1(x_{n-1})}{2}h\right]\beta \end{pmatrix},$$

其系数矩阵 A 是 $n-1$ 阶的三对角矩阵. 当 $\det A\neq 0$ 时,可以用追赶法求解. 设 $L=\max\limits_{a\leqslant x\leqslant b}|q_1(x)|$,当 $L\neq 0$,且步长 $h<\dfrac{2}{L}$时,(10.95)有唯一解;当 $L=0$ 时,对于任意的步长 h,(10.95)都收敛.

例 10.8.2 用有限差分方法求边值问题

$$y''(x)=\frac{x}{1+x^2}y'(x)+\frac{3}{1+x^2}y(x)+\frac{6x-3}{1+x^2},x\in[0,1],y(0)=1,y(1)=2$$

的数值解,将[0,1]平均分成六等份.

解 (1) 因为 $q_1(x)=\dfrac{x}{1+x^2},q_2(x)=\dfrac{3}{1+x^2},q_3(x)=\dfrac{6x-3}{1+x^2}$,则它内节点$x_k=hk$ $(k=1,2,3,4,5)$上的差分方程(10.95)的具体形式为

$$\begin{cases} -[2+h^2q_2(x_1)]y_1+\left[1-\dfrac{q_1(x_1)}{2}h\right]y_2=h^2q_3(x_1)-\left[1+\dfrac{q_1(x_1)}{2}h\right]\alpha, \\ \left[1+\dfrac{hx_k}{2(1+x_k^2)}\right]y_{k-1}-\left[2+\dfrac{3h^2}{1+x_k^2}\right]y_k+\left[1-\dfrac{hx_k}{2(1+x_k^2)}\right]y_{k+1}=h^2\dfrac{6x_k-3}{1+x_k^2}\quad(k=2,3,4), \\ \left[1+\dfrac{q_1(x_4)}{2}h\right]y_4-[2+h^2q_2(x_5)]y_5=h^2q_3(x_5)-\left[1-\dfrac{q_1(x_5)}{2}h\right]\beta. \end{cases}$$

将 $\alpha=1,\beta=2,n=6,h=\dfrac{1}{6}$代入(10.101)得到方程组

$$\begin{pmatrix} -2.0833 & 1.0000 & 0 & 0 & 0 \\ 1.0000 & -2.0811 & 0.9865 & 0 & 0 \\ 0 & 1.0135 & -2.0750 & 0.9750 & 0 \\ 0 & 0 & 1.0250 & -2.0667 & 0.9667 \\ 0 & 0 & 0 & 1.0333 & -2.0577 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}=\begin{pmatrix} -1.0833 \\ -0.0541 \\ -0.0250 \\ 0 \\ -2.0577 \end{pmatrix}.$$

算出数值解

$$y_1=1.0478, y_2=1.0996, y_3=1.2027, y_4=1.3910, y_5=1.6985,$$

其精确解 $y=1+x^3$ 的对应值为

$$y(x_1)=1.0046, y(x_2)=1.0370, y(x_3)=1.1250, y_4=1.2963, y_5=1.5787.$$

(二) 常微分方程边值问题的有限差分方法的 MATLAB 程序

现提供用有限差分方法求解线性边值问题(10.95)数值解的 MATLAB 主程序如下：

用有限差分方法求解线性边值问题(10.95)数值解的 MATLAB 主程序

输入量：q1，q2 和 q3 分别是 $y''=q_1(x)y'+q_2(x)y+q_3(x)$ 中 $q_1(x)$，$q_2(x)$，$q_3(x)$ 对应的 M 函数文件，alpha，beta 分别是边值条件 $y(a)=\alpha$，$y(b)=\beta$；a 和 b 是自变量 x 所在的区间 $[a,b]$ 的端点，h 是步长.

输出量：向量 $\boldsymbol{X}$ 的元素是自变量 x 的取值，向量 $\boldsymbol{Y}$ 的元素是用有限差分方法求出线性边值问题(10.95)在向量 $\boldsymbol{X}$ 的元处的数值解，k 是自变量 x 取值的序号，wucha(k+1)=norm(y(k+1)-y(k))，$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 分别是(10.101)的系数矩阵和常数向量.

```
function [k,A,B1,X,Y,y,wucha,p] = yxcf(q1,q2,q3,a,b,alpha,beta,h)
n = fix((b - a)/h); X = zeros(n+1,1);
Y = zeros(n+1,1); A1 = zeros(n,n);
A2 = zeros(n,n); A3 = zeros(n,n); A = zeros(n,n);B = zeros(n,1);
for k = 1:n
X = a:h:b;
k1(k) = feval(q1,X(k)); A1(k+1,k) = 1 + h*k1(k)/2;
k2(k) = feval(q2,X(k));
A2(k,k) = -2 - (h.^2)*k2(k);
A3(k,k+1) = 1 - h*k1(k)/2; k3(k) = feval(q3,X(k));
end
for k = 2:n
B(k,1) = (h.^2)*k3(k);
end
B(1,1) = (h.^2)*k3(1) - (1 + h*k1(1)/2)*alpha;
B(n-1,1) = (h.^2)*k3(n-1) - (1 + h*k1(n-1)/2)*beta;
A = A1(1:n-1,1:n-1) + A2(1:n-1,1:n-1) + A3(1:n-1,1:n-1);
B1 = B(1:n-1,1);
Y = A\B1;Y1 = Y'; y = [alpha;Y;beta];
```

```
for k = 2:n + 1
wucha(k) = norm(y(k) - y(k - 1)); k = k + 1;
end
X = X(1:n + 1); y = y(1:n + 1,1); k = 1:n + 1;
wucha = wucha(1:k,:); plot(X,y(:,1),'mp')
xlabel('轴 \it x'); ylabel('轴 \it y'),legend('是边值问题的数值解 y(x)
的曲线')
title('用有限差分法求线性边值问题的数值解的图形'),
p = [k',X',y,wucha'];
```

例 10.8.3 用有限差分方法求边值问题

$$y''(x) = \frac{x}{1+x^2}y'(x) + \frac{3}{1+x^2}y(x) + \frac{6x-3}{1+x^2}, x \in [0,1], y(0) = 1, y(1) = 2$$

的数值解,将[0,1]分别平均分成 $n = 6$ 等份和 $n = 36$ 等份,说明 n 对数值解的误差的影响,并与精确解比较,画出它们的图形.

解 (1) 根据 $q_1(x) = \frac{x}{1+x^2}, q_2(x) = \frac{3}{1+x^2}, q_3(x) = \frac{6x-3}{1+x^2}$,建立对应的 M 函数文件

```
function y = q1(x)
y = x/(1 + x^2);
function y = q2(x)
y = 3/(1 + x^2);
function y = q3(x)
y = (6 * x - 3)/(1 + x^2);
```

(2) $n = 6$ 时,输入程序

```
>> n = 6;a = 0;b = 1;alpha = 1;beta = 2;h = (b - a)/n;
[k,A,B,X,Y,y,wucha,p] = yxcf(@ q1,@ q2,@ q3,a,b,alpha,beta,h),
x = 0:h:1;y1 = 1 + x.^3,wu - y1' - y; [k',X',y,y1',wucha',wu],hold on
plot(x,y1,'bo')
legend('边值问题的数值解 y(x)的曲线','边值问题的精确解 y(x)的曲线')
title('n = 6,用有限差分法求线性边值问题的数值解及其精确解的图形')
hold off
```

将运行后屏幕显示 $n = 6$ 时的计算结果列入表 10 - 21 中,图形见图 10 - 36. 从图 10 - 36 可以看出,此时除了端点的值的以外,数值解与精确解的绝对误差在 0.12 与 0.03 之间.

表 10－21 例 10.8.3 的计算结果

k	自变量 X	数值解 y	精确解 y_1	$wucha = \lvert y(k+1) - y(k) \rvert$	$wu = y_1 - y$
1	0	1.000 0	1.000 0	0	0
2	0.166 7	1.047 8	1.004 6	0.047 8	－0.043 2
3	0.333 3	1.099 6	1.037 0	0.051 8	－0.062 5
4	0.500 0	1.202 7	1.125 0	0.103 1	－0.077 7
5	0.666 7	1.391 0	1.296 3	0.188 3	－0.094 7
6	0.833 3	1.698 5	1.578 7	0.307 5	－0.119 8
7	1.000 0	2.000 0	2.000 0	0.301 5	0

取 $n=36$ 时的计算结果(略),图形见图 10－37,从图 10－37 可以看出,此时数值解与精确解的图形几乎重合,说明它们的误差已经相当小. 比较图10－36和图 10－37 可见,n 越大,数值解与精确解的误差越小.

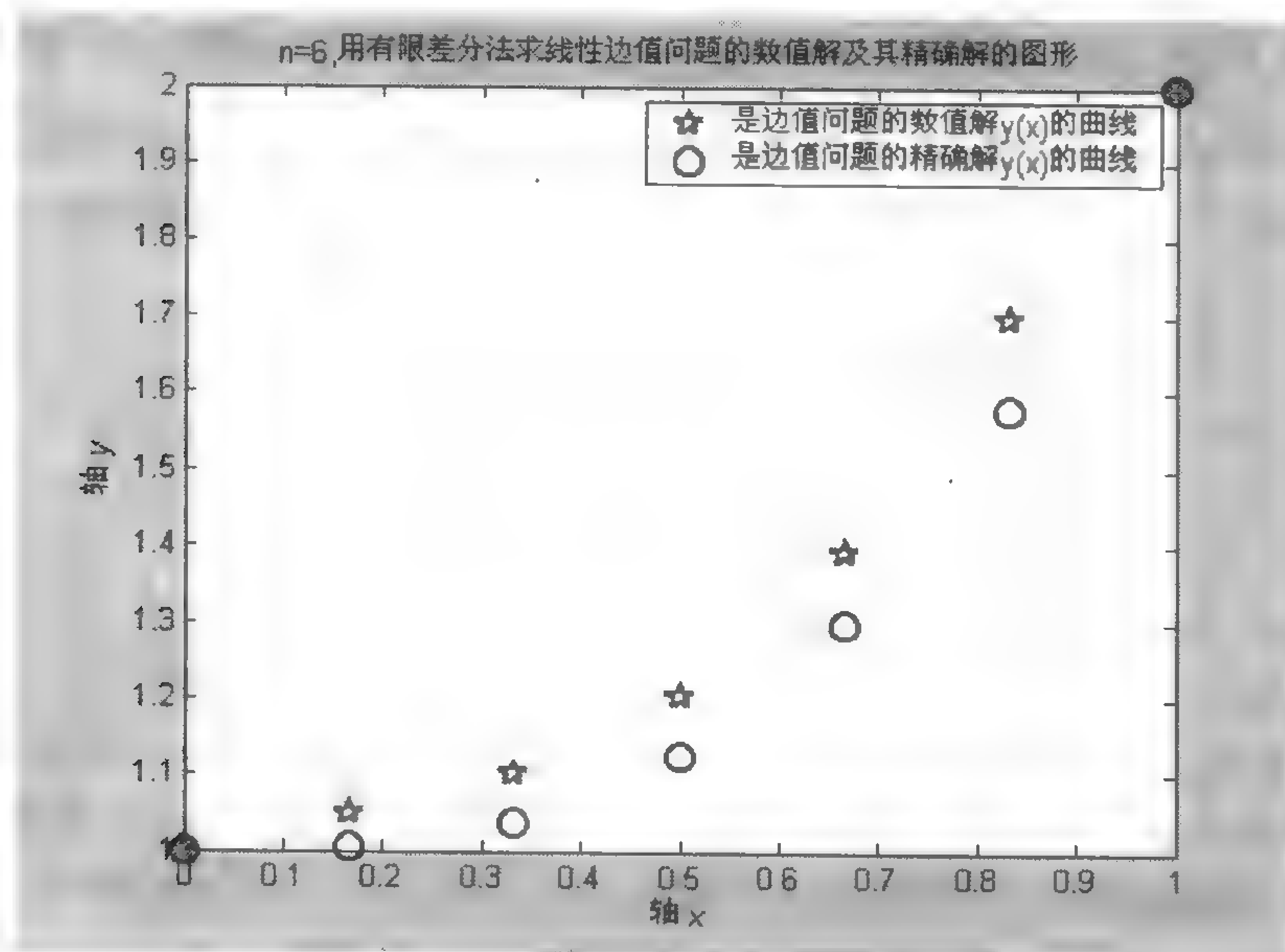


图 10－36 $n=6$ 时数值解和精确解的图形

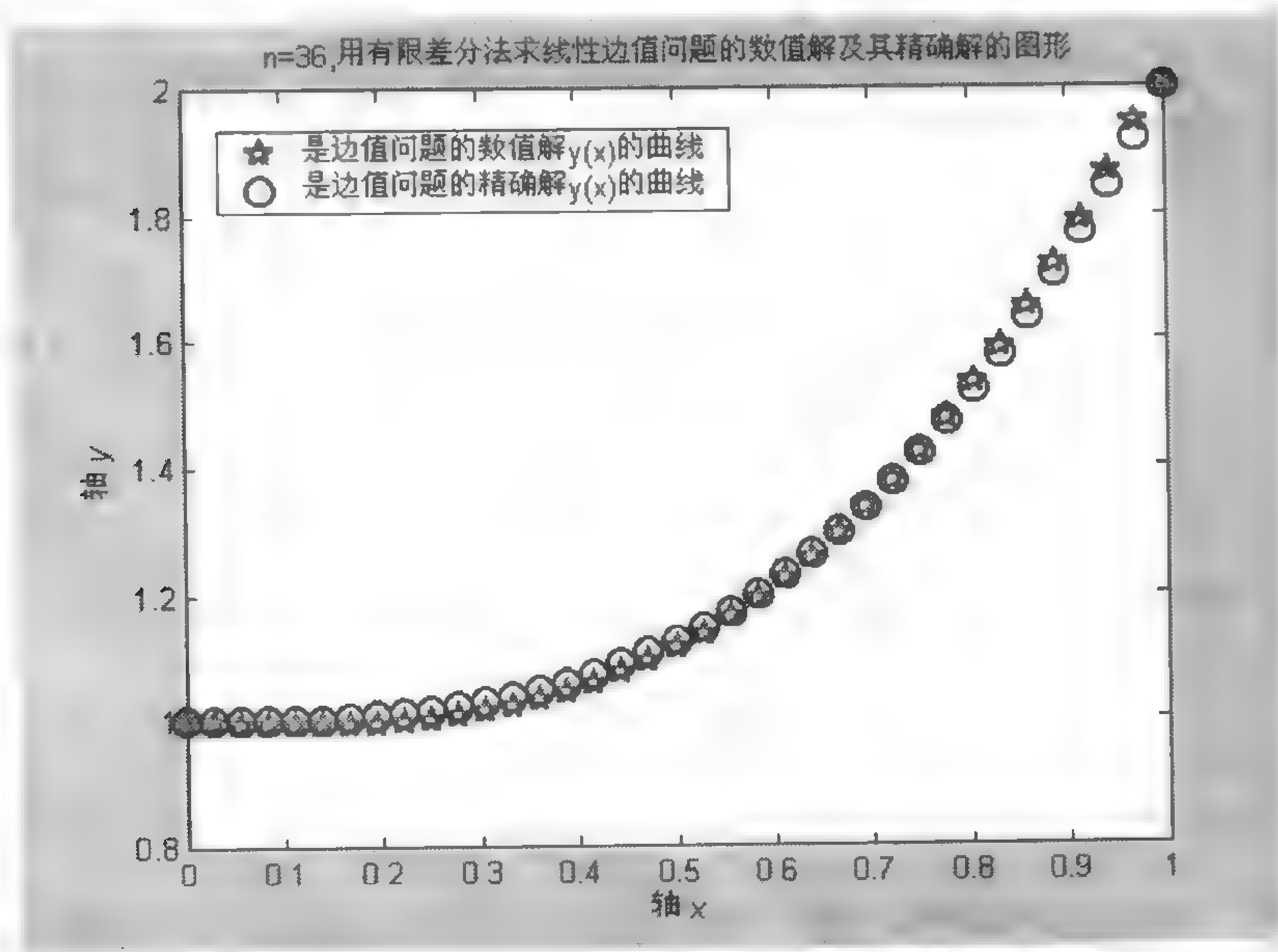


图 10－37　$n=36$ 时数值解和精确解的图形

10.8.3　求解常微分方程(组)边值问题数值解的 MATLAB 库函数

MATLAB 软件提供了一套求解一阶常微分方程(组)边值问题数值解的程序 bvp4c.m,称其为库函数 bvp4c. 它可以解决三种类型的边值问题,分别介绍如下.

(一) 一阶常微分方程(组)一般的两点边值问题

bvp4c 可以求解一阶常微分方程(组)一般的两点边值问题

$$\begin{cases} y'=f(x,y), x\in[a,b], \\ \varphi(y(a),y(b))=0, \text{一般的两点边界条件} \end{cases} \tag{10.102}$$

的数值解. bvp4 产生的数值解 $s(x)$ 在 $[a,b]$ 上具有一阶连续导数. bvp4 利用配置函数 bvpinit 和 deval 等求解. 即先用网格点将整个区间 $[a,b]$ 分成若干个子区间,将配置条件施加到所有子区间上,通过求解代数方程组解出数值解,然后在每一个子区间上估计数值解的误差. 如果此解不满足误差要求,则解函数调整网格. 这样用户首先需要用配置函数 bvpinit 提供初始网格点及在相应网格点上精确解的初始逼近,然后用函数 bvp4 和配置函数 deval 输出数值解.它调用格式如下:

调用格式: solinit = bvpinit(X,YINIT)或 solinit = bvpinit

(X,@YINITfun)

```
SOL = bvp4c (@ODEFUN,@BCFUN,SOLINIT)        (*)
  xint =[x1,x2,…;xn] 或 X = linspace(a,b,n);
YINT = deval(sol,xint)
```

其中输入量:(1) ODEFUN 是描述函数 $f(x,y)$ 的函数,一般为 M 函数文件形式;

(2) BCFUN 是描述一般的两点边界条件 $\varphi(y(a),y(b))=0$ 的函数,一般为 M 函数文件形式;

(3) SOLINIT 是被指定为 x 和 y 域的结构. x 是初始网格点,两个边界条件被固定为 SOLINIT. x(1) = a, SOLINIT.x(end) = b. y 表示在节点 SOLINIT.x(i)处对 y(x(i))的初始猜测解 SOLINIT. y(:, i),一般用函数 bvpinit实现.bvpinit 的调用格式为

solinit = bvpinit(X,YINIT) 或 solinit = bvpinit(X,@Yfun)

输入 X 的格式有两种:X = linspace(a,b,10)或 X = [X(1),X(2),…; X(end)],其中 $X(1)<X(2)<\cdots<X(\text{end})$.

输入 YINIT 的格式有两种:向量或函数.

如果 YINIT 是以向量的格式输入时,则 YINIT(i)是在 X 中所有的网格点处的精确解的第 i 个分量 $Y(i,:)$ 的猜测值.

如果 YINIT 是 x 的函数 *yfun* ,则用格式 solinit = bvpinit(x, @yfun),对于 $[a,b]$ 上的任意 x, $yfun(x)$ 返回解 $y(x)$ 的猜测值.

输出量:(1) SOL 是如下形式的结构

SOL.x——由函数 bvp4 生成的网格点;

SOL.y——在网格点 SOL.x 处对精确解 $y(x)$ 的数值解 $s(x)$ 的值;

SOL.yp——在网格点 SOL.x 处对 $y(x)$ 的一阶导数 $y'(x)$ 的近似值;

SOL.solver——'bvp4c'

(2) 因为 SOL 不能直接输出数值解,所以用 bvp4 的配置函数 deval 估计微分方程问题在向量 XINT 中所有分量处的解 *YINT*,其调用格式为

YINT = deval (SOL,XINT).

例 10.8.4 求微分方程 $y''+|y|=0$ 在区间 $[0,4]$ 上满足边界条件: $y(0)=0$ 和 $y(4)=-2$ 的数值解,并画出 y 和 y' 数值解的图形.

解 (1) 首先将问题写成一阶微分方程组边值问题 $\begin{cases} y_1'=y_2, \\ y_2'=-|y_1| \end{cases}$ 在区间 $[0,4]$ 上满足边界条件: $y_1(0)=0$ 和 $y_1(4)=-2$.

(2) 建立名为 odefun1.m 和 bcfun1.m 的两个 M 函数文件

```
function dydx = odefun1(x,y)
```

```
dydx =[y(2); -abs(y(1))];
function res =bcfun1(ya,yb)
res =[ya(1);yb(1) +2];
```

(3) 求数值解,在 MATLAB 工作窗口输入程序

```
> >solinit =bvpinit([0 1 2 3 4],[1 0]);
sol =bvp4c(@odefun1,@bcfun1,solinit);
xint =linspace(0,4,35)
yint =deval(sol,xint)
plot(xint, yint (1,:),'bo',xint, yint (2,:) ,'rp')
xlabel('轴\it x'); ylabel('轴\it y')
legend('边值问题的数值解 y(x)的曲线','边值问题中 y(x)的导数 dy(x)/dx
的数值解的曲线')
title('用一阶常微分方程组的两点边值问题求二阶常微分方程数值解的图形')
```

运行后屏幕显示计算数值解的结果和相关信息及其图形(见图 10-38)如下

```
sol =
         x: [1x22 double]
         y: [2x22 double]
        yp: [2x22 double]
    solver: 'bvp4c'
xint =
   Columns 1 through 7
   0   0.1176   0.2353   0.3529   0.4706   0.5882   0.7059
..................................
   Columns 29 through 35
  3.2941   3.4118   3.5294   3.6471   3.7647   3.8824   4.0000
yint =
   Columns 1 through 7
   0   0.2426   0.4819   0.7144   0.9371   1.1468   1.3406
  2.0669   2.0526   2.0099   1.9395   1.8422   1.7194   1.5728
   Columns 8 through 14
..................................
   Columns 29 through 35
   -0.3163   -0.5650   -0.8216   -1.0896   -1.3726
-1.6747   -2.0000   -2.0910   -2.1427   -2.2241   -2.3365   -2.4811
-2.6601   -2.8761
```

说明:bvp4c 调用格式中的(*)也可换成

```
SOL = bvp4c (@ODEFUN,@BCFUN,SOLINIT,OPTIONS)
```

或

```
SOL - bvp4c (@ODEFUN,@BCFUN,SOLINIT,OPTIONS,P1,P2,…)
```

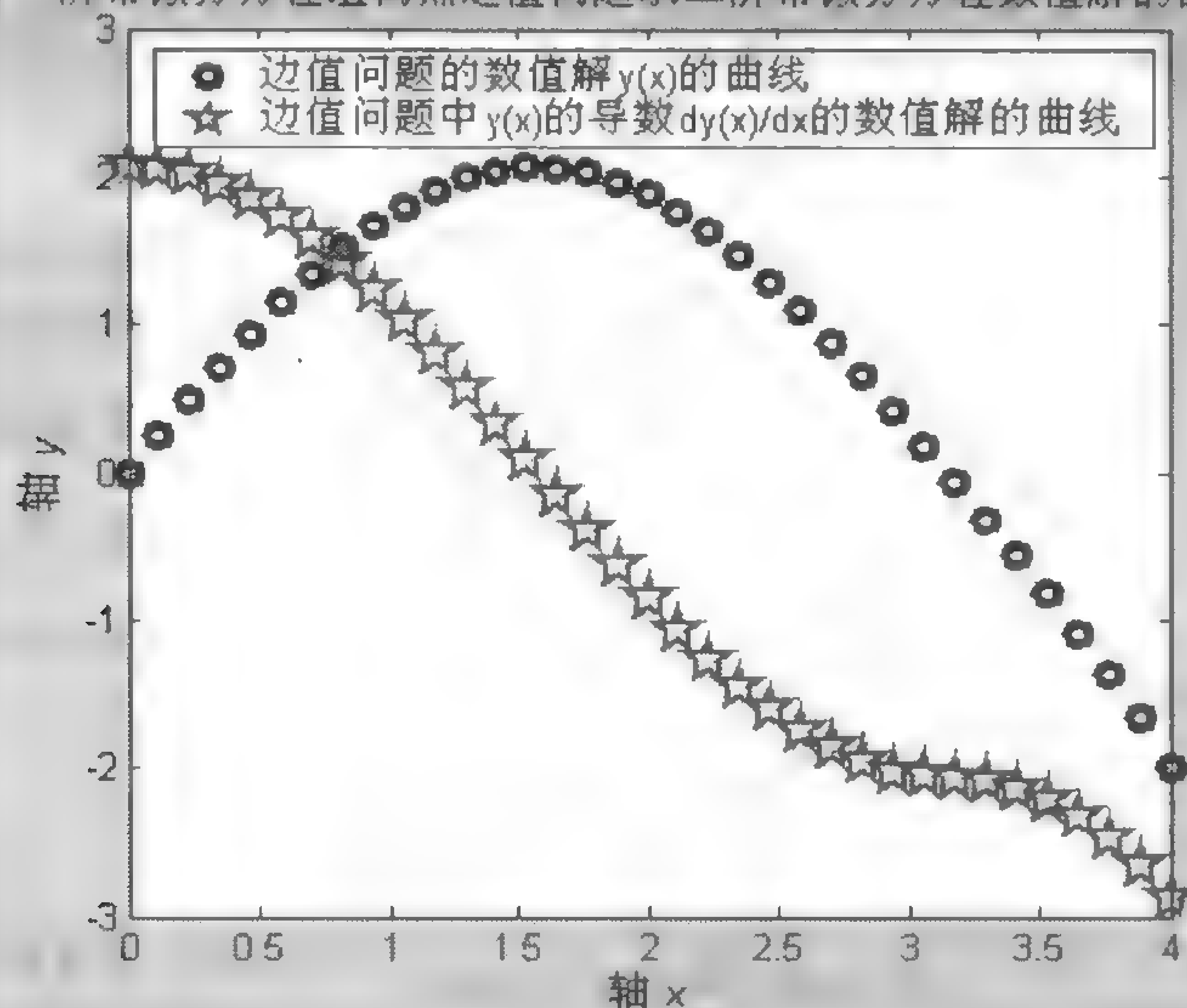


图 10－38　$y''+|y|=0$ 在 $[0,4]$ 上满足 $y(0)=0$ 和 $y(4)=-2$ 的数值解

其中 OPTIONS 是 bvp4c 可选项设置的结构，它的数据类型为结构数组．用户可以通过这些选项的设置改变 bvp4c 中缺省值的设置，提高效率．用 bvpset 设置 OPTIONS 结构的选项，调用格式为

```
OPTIONS = bvpset('name1','value1','name2','value2',...)
```

其中 name1，value2 是属性名；value1，value2 是属性值．常用的属性名如表 10－22 所示：

表 10－22　BVP 的 OPTIONS 结构的属性名和属性值

属性名	属性值	属性说明	属性类型
restol	1e－3	用于所有残量限制的相对误差	用 bvp4c 解出的数值解 $S(x)$ 满足 $S'(x)=f(x,S(x))+res(x)$，$res(x)$为残差．在网格的每一个子区间上，解的第 i 个分量的残差的分量 $res(i)$，满足容差条件 $\left\| res(i) \middle/ \max\left\{ \lvert f(i) \rvert, \frac{abstol(i)}{restol} \right\} \right\| \leqslant reltol$
abstol	1e－6	用于所有残量限制的绝对误差	

续表

属性名	属性值	属性说明	属性类型
FJacobian	函数	(1) 计算$\frac{\partial f(x,y)}{\partial y}$,使用格式 dfdy = fjac(x,y) (2) 若有未知参数 p,使用格式 [dfdy,dfdp] = fjac(x,y,p)	在缺省值的情况下,bvp4c 用有限差分逼近所有的偏导数.如果用户提供解析偏导数$\frac{\partial f}{\partial y}$,$\frac{\partial bc}{\partial ya}$,$\frac{\partial bc}{\partial yb}$,则 bvp4c 可能更有效.如果有未知参数 p,用户可以提供偏导数$\frac{\partial f}{\partial p}$,$\frac{\partial b\varphi}{\partial p}$
BCJacobian	函数	(1) 求 $\varphi(ya,yb)$偏导数,使用格式 [dbdya, dbdyb] = bcjac(ya,yb) (2) 若有未知参数 p,使用格式 [dbdya, dbdybdbdp] = bcjac(ya,yb,p)	
Nmax	floor {1000/n}	允许的最大网点数,在缺省时限定代数方程组为 1000 个方程	bvp4c 通过解代数方程组来决定 BVP 在网点处的解.此代数方程组的规模与微分方程的个数 n 和当前的网点数有关.当被允许使用的网点用完时,计算结束.此时,若 bvp4c 列出警告信息,则说明当前的计算没有满足误差要求.但是它提供了一个很好的初始猜测,这样可以放宽误差要求或增加网点数 Nmax 重新开始计算
stats	on/off	指定关于数值解的统计资料.若选 on,则列出最终的网点数目,解的最大残余量,调用 odefun 估计 $f(x,y)$的次数,调用 bcfun 估计 $\varphi(ya,yb)$的次数	

(二) 含未知参数的一阶常微分方程(组)的一般两点边值问题

bvp4c 还可以求解包含未知参数向量 $\boldsymbol{P}$ 的一阶常微分方程(组)一般的两点边值问题

$$\begin{cases} \boldsymbol{y}'(x)=\boldsymbol{f}(x,\boldsymbol{y}(x),\boldsymbol{P}),x\in[a,b], \\ \varphi(y(a),y(b),\boldsymbol{P})=0,\text{一般的两点边界条件} \end{cases} \tag{10.103}$$

的解 $y(x)$的数值解.在这种条件下,边界条件的个数应保证能够解出此数值解和参数向量 $\boldsymbol{P}$.它的调用格式如下:

调用格式: solinit = bvpinit(X, YINIT, P) 或 solinit = bvpinit(X,@YINITfun,P)

```
SOL = bvp4c (@ODEFUN,@BCFUN,SOLINIT)    (**)
xint =[x1,x2,…;xn]   或  X = linspace(a,b,n);
```

```
YINT = deval(sol,xint)
```

其中**输入量**:在 solinit 中必须提供未知参数 P 的猜测值,在 ODEFUN 和 BCFUN及 OPTIONS 等的 M 函数文件的输入量中需要加入参数 P;其他同上.

输出量:参数 P 估计值,其他同上.

说明:bvp4c 调用格式中的(* *)也可换成

```
SOL = bvp4c (@ODEFUN,@BCFUN,SOLINIT,OPTIONS)
```

或

```
SOL = bvp4c (@ODEFUN,@BCFUN,SOLINIT,OPTIONS,P1,P2,...)
```

例 10.8.5 求微分方程组 $\begin{cases} y'=z, \\ z'=-(\lambda-2q\cos 2x)y \end{cases}$ 在区间 $[0,\pi]$ 上满足边界条件:$y(0)=1$,$z(0)=0$ 和 $z(\pi)=0$ 的数值解,取 $q=5$,并画出 y 和 z 数值解的图形.

解 (1) 建立名为 odefun2.m 和 bcfun2.m 的两个 M 文件

```
function  dydx = odefun2(x,y,lamda)
q = 5;dy(1) = y(2);dy(2) = -(lamda - q * 2 * cos(2 * x)) * y(1);
dydx = [dy(1);dy(2)];
function res = bcfun2(ya,yb,lamda)
res = [ ya(1) -1;ya(2);yb(2)];
function yinit = init2(x)
yinit = [cos(4 * x); -4 * sin(4 * x)];
```

(2) 调用 dzdx1.m 求解,在 MATLAB 工作窗口输入程序

```
>> lamda = 2; solinit = bvpinit(0:pi/32:pi,@init2,lamda)
sol = bvp4c(@odefun2,@bcfun2,solinit)
xint = 0:pi/32: pi
YINT = deval(sol,xint)
plot(xint,YINT(1,:),'bo',xint,YINT(2,:),'rp')
xlabel('轴\it x'); ylabel('轴\it y')
legend('是方程组解 y 的曲线','是方程组解 z 的曲线')
title('用含未知参数的一阶常微分方程组的两点边值问题求二阶常微分方程组数值解的图形')
```

运行后屏幕显示计算数值解和参数 λ 的结果和相关信息及其图形(见图 10-39)如下

```
solinit =
              x: [1x33 double]
              y: [2x33 double]
     parameters: 2
sol =
              x: [1x63 double]
```

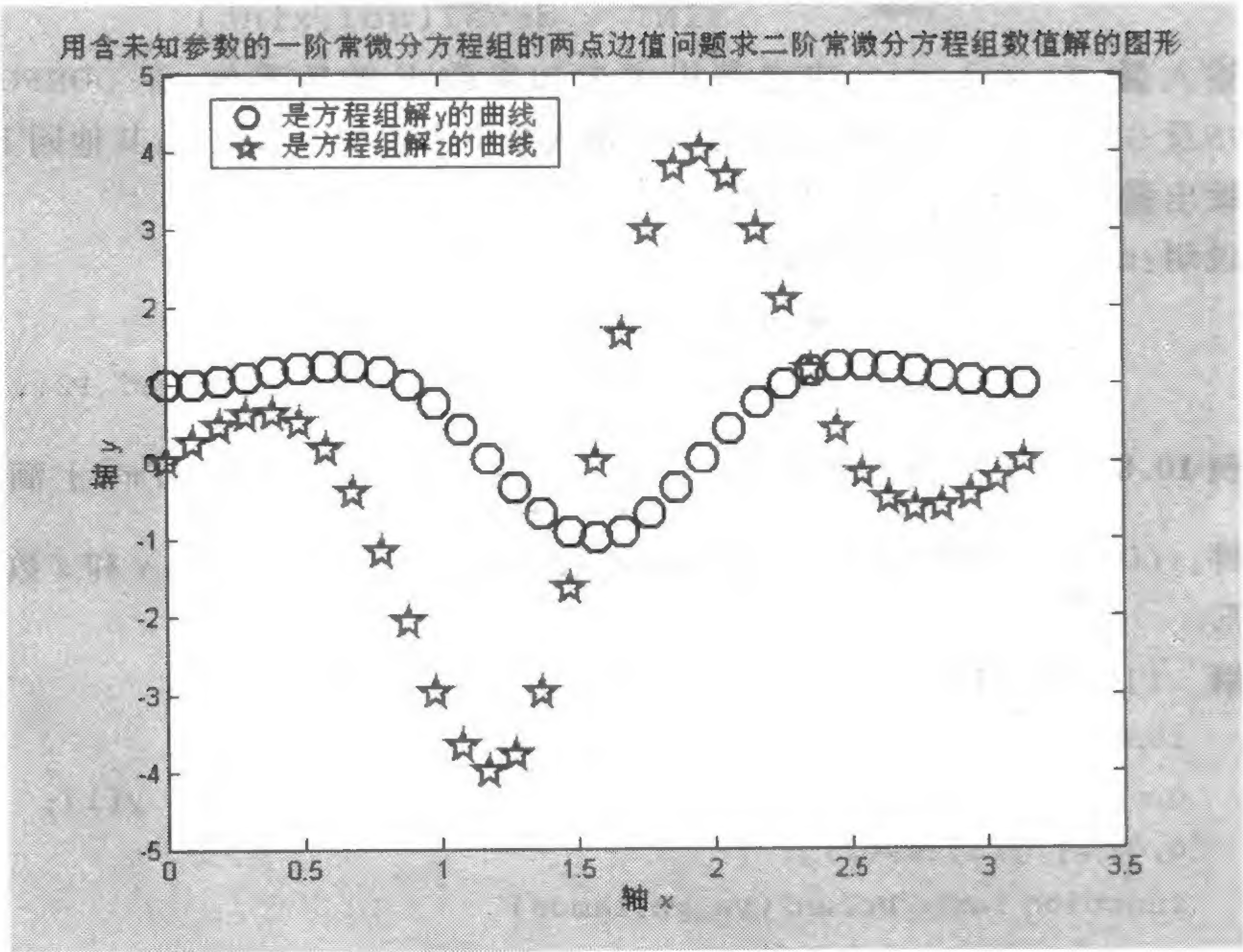


图 10－39

```
            y: [2x63 double]
           yp: [2x63 double]
       solver: 'bvp4c'
   parameters: 7.4491
xint =
   Columns 1 through 7
       0   0.0982   0.1963   0.2945   0.3927   0.4909   0.5890
       ..........................................................
   Columns 29 through 33
     2.7489     2.8471     2.9452     3.0434     3.1416
  YINT =
   Columns 1 through 7
   1.0000   1.0122   1.0471   1.0997   1.1611   1.2179   1.2522
        0   0.2451   0.4574   0.5997   0.6289   0.4978   0.1640
   Columns 8 through 14
        ........................................................
   Columns 29 through 33
     1.1611     1.0997     1.0471     1.0122     1.0000
```

```
-0.6289   -0.5997   -0.4574   -0.2451       0
```

例 10.8.6 求微分方程 $ay''+xy'=a\pi^2\cos\pi x-\pi x\sin\pi x$ 在区间 $[-1,1]$ 上满足边界条件：$y(-1)=-2, y(1)=0$ 和 $0<a\ll 1$ 的数值解，并画出 y 和 y' 数值解的图形.

解 （1）首先将问题写成一阶微分方程组边值问题

$$\begin{cases} y_1'=y_2, \\ y_2'=\dfrac{1}{a}[-xy_2+a\pi^2\cos\pi x-\pi x\sin\pi x] \end{cases}$$

在 $[-1,1]$ 上满足边界条件：$y_1(-1)=-2, y_1(1)=0$ 和未知参数 a 在 $0<a\ll 1$.

（2）建立名为 odefun3.m 和 bcfun3.m，fjac.m 和 bcjac.m 的四个 M 文件

```
function dydx = odefun3(x,y,a)
pix = pi * x;
dydx = [y(2,:); -x/a.* y(2,:) + pi^2 * cos(pi * x) - pi * x/a.*
sin(pi * x)];
function res = bcfun3(ya,yb,a)
res = [ya(1) + 2;yb(1)];
function jac = fjac(x,y,a)
jac = [0  1;0  -x/a]; % 估计 ODE 函数的雅可比行列式
function [dBCdya,dBCdyb] = bcjac(ya,yb,a)
dBCdya = [1 0;0 0]; dBCdyb = [0 0;1 0]; % 估计边界条件的偏导数
```

（3）求数值解，在 MATLAB 工作窗口输入程序

```
>> options = bvpset('FJacobian',@fjac,'BCJacobian',@bcjac,
'Vectorized','on');
sol = bvpinit([-1 -0.5 0 0.5 1],[1 0]);
a = 0.1; for i =2:4
  a =a/10;  sol = bvp4c(@odefun3,@bcfun3,sol,options,a);
end
figure; plot(sol.x,sol.y(1,:));grid
axis([-1 1 -2.2 2.2]);
title('在 x=0 处有一个突变');
xlabel('x');
ylabel('solution y')
```

运行后屏幕显示计算数值解的结果和相关信息（略）及其图形（见图 10-40）：

（三）含未知参数和奇异点的一阶常微分方程（组）的一般两点边值问题

bvp4c 还可以求解包含未知参数向量 $\boldsymbol{P}$ 的一阶常微分方程（组）一般的两点边值问题

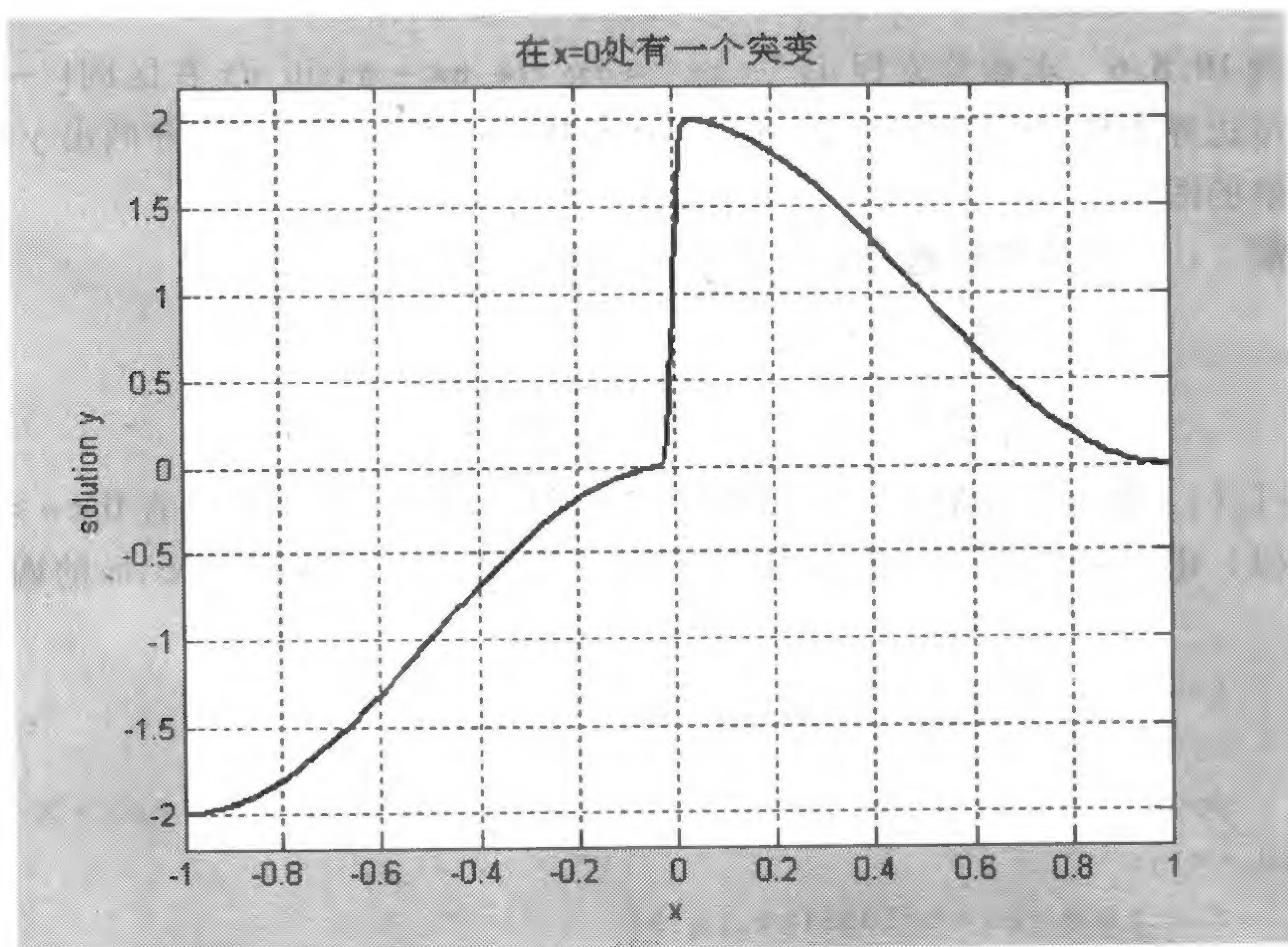


图 10－40

$$\begin{cases} \boldsymbol{y}'(x)=\dfrac{\boldsymbol{S}\cdot \boldsymbol{y}}{x}+\boldsymbol{f}(x,\boldsymbol{y}(x),\boldsymbol{P}),x\in[0,b],b>0, \\ \varphi(y(a),y(b),\boldsymbol{P})=0,\text{一般的两点边界条件} \\ S\cdot y(0)=0 \end{cases} \quad (10.104)$$

的解 $y(x)$ 的数值解，调用格式与(二)类似，这里不再叙述.



习 题 10.8

1. 用线性打靶法求线性边值问题

$y''(x)=\dfrac{c_1x}{1+x^2}y'(x)-\dfrac{c_2}{1+x^2}y(x)+c_3,x\in[0,2],y(0)=1.25,y(2)=-0.25,c_1=c_2=3,c_3=0.6$ 的数值解，$h=0.2$，并与精确解比较，画出它们的图形.

2. 用有限差分方法求边值问题

$y''(x)=\dfrac{2x}{1+x^2}y'(x)+\dfrac{4}{1+x^2}y(x)+\dfrac{5x-3}{1+x^2},x\in[0,1],y(0)=1,y(1)=2$ 的数值解，将 $[0,1]$ 平均分成六等份.

3. 用有限差分方法求边值问题

$y''(x)=\frac{x}{1+x^2}y'(x)+\frac{3}{1+x^2}y(x)+\frac{6x-3}{1+x^2}, x\in[0,1], y(0)=1, y(1)=2$ 的数值解，将 $[0,1]$ 分别平均分成 $n=5$ 等份和 $n=35$ 等份，说明 n 对数值解的误差的影响，并与精确解比较，画出它们的图形.

4. 求解马蒂厄(Mathieu)方程 $y''+(\lambda-2q\cos 2x)y=0$ 及参数 λ，在区间 $[0,\pi]$ 上满足边界条件：$y(0)=1, y'(0)=0$ 和 $y'(\pi)=0$ 的数值解，取 $q=5$，并画出 y 和 y' 数值解的图形.

5. 求微分方程 $y''+2|y|=0$ 在区间 $[0,4]$ 上满足边界条件：$y(0)=0$ 和 $y(4)=-2$ 的数值解，并画出 y 和 y' 数值解的图形.

6. 求微分方程 $ay''+3xy'=a\pi^2\cos\pi x-2\pi x\sin\pi x$ 在区间 $[-1,1]$ 上满足边界条件：$y(-1)=-2, y(1)=0$ 和 $0<a\ll 1$ 的数值解，并画出 y 和 y' 数值解的图形.

7. 用线性有限差分方法求

$$\begin{cases} y''=\left(\frac{-4}{x}\right)y'-\left(\frac{2}{x^2}\right)y+\left(\frac{2}{x^2}\right)\ln x, 1\leqslant x\leqslant 2, \\ y(1)=\frac{1}{2}, y(2)=\ln 2 \end{cases}$$

的数值解.

8. 求 $\begin{cases} y''+y=0, 0\leqslant x\leqslant\pi, \\ y(0)=1, y(\pi)=-1; h=\frac{\pi}{3} \end{cases}$ 的数值解.

9. 求 $\begin{cases} y''=\left(\frac{-2}{x}\right)y'+\left(\frac{2}{x^2}\right)y+\frac{\sin(\ln x)}{x^2}, 1\leqslant x\leqslant 2, \\ y(1)=1, y(2)=2 \end{cases}$ 的数值解.

10. 求线性边值问题 $\begin{cases} y''=\left(\frac{-2}{x}\right)y'+\left(\frac{2}{x^2}\right)y+\frac{\sin(\ln x)}{x^2}, 1\leqslant x\leqslant 2, \\ y(1)=1, y(2)=2 \end{cases}$ 的数值解.